Reduced models for the behavior of VLSI circuits

Article · January 1991

Source: OAI

CITATIONS
17

READS
1,027

1 author:

Arjan J Van Genderen
Delft University of Technology
49 PUBLICATIONS **545** CITATIONS

SEE PROFILE

REDUCED MODELS FOR THE BEHAVIOR OF VLSI CIRCUITS

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft, op gezag van de
Rector Magnificus, prof. drs. P.A. Schenck,
in het openbaar te verdedigen ten overstaan van een
commissie aangewezen door het College van Dekanen
op donderdag 3 oktober 1991 te 14.00 uur

door

Arie Johannes van Genderen

geboren te Vlaardingen
elektrotechnisch ingenieur

Dit proefschrift is goedgekeurd door de promotor
prof. dr. ir. P.M. Dewilde

Published and distributed by:

Delft University Press
Stevinweg 1
2628 CN Delft
The Netherlands

Telephone +31 15 783254
Fax +31 15 781661

ISBN 90-6275-721-9 / CIP

Printed in The Netherlands.

1. INTRODUCTION

1.1 The verification of the behavior of integrated circuits

Modern integrated circuits may contain over 1,000,000 transistors, and approximately the same number of interconnections between these transistors. To manage the complexity of their design, designers must resort to computer aids (Computer-Aided Design). One point where the CAD programs are indispensable is the verification of the behavior of these circuits. In addition to obvious errors such as wrong or incomplete connections between different parts of the circuit, electrical errors may occur because of physical limitations such as too great a drop in voltage along supply lines, too large delay time values, and unintended capacitive coupling effects between different interconnections. With the aid of special purpose CAD programs that contain methods to model, verify and/or simulate these effects, it is possible to accurately and efficiently verify an integrated circuit for these types of errors. As a consequence, the design time of the integrated circuit is decreased, while the likelihood of the circuit working correctly when it has been fabricated is increased.

The verification of the physical behavior of integrated circuits, can be divided in two steps as shown in Figure 1.1. During the first step, a description of the electrical circuit is reconstructed from the layout description of the circuit. The electrical circuit description contains the transistors that are present in the circuit and information about the interconnections between them. By using technology information such as sheet-conductivity and geometrical dimensions of the interconnect wires, realistic values for the interconnect resistances and the interconnect capacitances are obtained. Then, during a second step, the behavior of the electrical circuit is verified by means of static verification or by means of simulation. With static verification an analysis performed on the circuit to check at least if all connections are present, and to verify if the values of the interconnect resistances and the values of interconnect capacitances are not too high. During simulation, a simulation model is chosen for the electrical circuit, and the behavior of the circuit is verified by checking the response of the simulation model to some input stimuli that are applied to model. The simulation and verification results are used by the designer to possibly modify the layout description of the circuit. In that case, one or more new extraction and verification steps are executed, until the

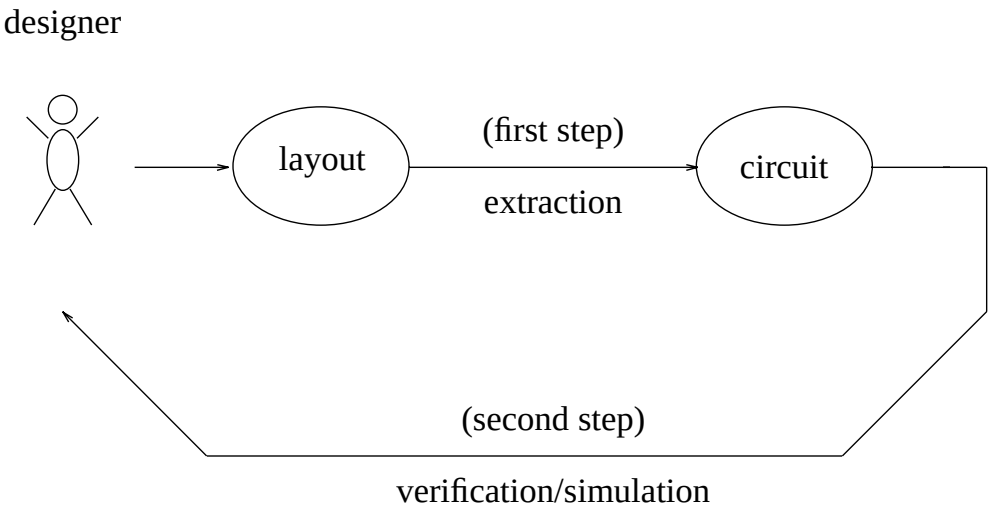


Figure 1.1. Verification of the behavior of integrated circuits.

designer is satisfied with the result.

For both verification steps mentioned above it is important that the models are reduced to a degree of complexity where they are simple enough to allow an efficient verification. In addition, the models should be accurate enough to adequatedly verify the behavior of the circuit.

1.2 An overview of this thesis

In this thesis we will focus on several aspects of the modeling of the behavior of integrated circuits for verification purposes. First, in Chapter 2, we will elaborate on the modeling of interconnect resistances in integrated circuits. In addition to the resistances, also the distribution of the interconnect capacitances over the resistance network is important to characterize the transmission behavior of the interconnections. Therefore this subject is also treated in Chapter 2.

In Chapter 3 we will discuss the modeling of three-dimensional capacitive effects between the interconnections of integrated circuits. The three-dimensional capacitive effects become more prominent as the horizontal dimensions of the circuit are scaled down, while the vertical dimensions are unchanged. In order to compute reliable capacitance values for the most critical parts of the circuit, an accurate yet efficient numerical technique is required that directly computes the capacitance values from the layout description of the circuit. A description of such a technique is given in Chapter 3.

Finally, in Chapter 4, we will describe a simulation model for quickly simulating the logic and timing behavior of large digital MOS circuits. Although the

simulation model that is presented in this chapter is not as accurate as the simulation model employed by a circuit simulator like SPICE, it provides useful information about resistance division effects, charge-sharing effects, delay times, spikes and races occurring in the circuit, and it can be used - unlike SPICE - to simulate on a workstation, in a reasonable amount of time, circuits containing over 100,000 transistors.

Although there are some overlap areas between the different subjects discussed in Chapter 2, 3 and 4, each of the chapters can be read independently.

2. RESISTANCE MODELING

2.1 Introduction

Resistances of interconnections in VLSI circuits play a significant role in the behavior of these circuits. For long poly-silicon and diffusion wires especially, the interconnect resistances may dominate over the impedances of the transistors that are connected to the wires. This will increase delay times as compared to situations where resistances can be neglected and, in certain circumstances, may cause incorrect functioning of the circuit. Therefore, in many cases it is required to calculate the resistances of the interconnections of integrated circuits from the mask layout information in order to ascertain correct circuit behavior. Based upon the resistances and based upon the characteristics of other circuit components (e.g. capacitances and transistors), delay times can accurately be estimated with the aid of some verification tool (e.g. the circuit simulator SPICE [1]) and the correct behavior of the circuit can be verified before the circuit is fabricated.

Several methods to calculate resistances of IC interconnections are known [2, 3, 4, 5, 6, 7, 8, 9, 10, 11]. Important requirements for the method to be chosen are:

1. It should be general enough to handle all interconnection geometries (including non-orthogonal geometries) occurring in VLSI circuits.
2. Because of process tolerances and requirements on the accuracy of delay times, it should be possible to obtain resistance values with an accuracy of approximately 5-15 %.
3. The method should be efficient with respect to computation time and memory usage, also for large interconnections.
4. The method should easily be incorporated in a CAD computer program to extract interconnect resistances from the mask layout specification.

An important side-problem of the modeling of interconnect resistance is the distribution of the interconnect capacitances over the resistance network. When neglecting the inductive effects, the transmission behavior of the IC interconnections is determined by their resistive effects and by their distributed capacitive effects. The latter effects can, in principle, be modeled by breaking up

the interconnections into pieces and connecting the distributed capacitances as lumped capacitances to the nodes of the lumped resistance network [12, 13]. In order to obtain accurate results, the interconnect capacitances should be distributed over the nodes of the lumped resistance network such that the distributed RC properties of the interconnection are accurately reflected in the RC model. Methods that allow the computation of such a distribution of the interconnect capacitances over the nodes of the resistance network have, for example, been described in [14, 15] and [16]. The number of nodes and elements in the final RC model should not be too high, in order to allow efficient verification afterwards, and the model should possess a transmission behavior that closely matches the transmission behavior of the original interconnect.

The contents of this chapter are arranged as follows. Section 2.2 treats the calculation of resistances of interconnection wires. Section 2.3 discusses the distribution of the interconnect capacitances over the terminals of the interconnections. Section 2.4 presents a solution scheme for the methods chosen in 2.2 and 2.3, and Section 2.5 gives the application of these techniques in a practical layout-to-circuit extraction program.

2.2 Resistance calculation

2.2.1 Introduction

For resistance calculation, the interconnections are approximated by two-dimensional homogeneous regions - corresponding to metal, poly-silicon and diffusion areas - that are connected to each other along surfaces representing the vias between these regions (see Figure 2.1). The terminals between which the resistances are calculated are given by the connections between the interconnections and the gates, drains and sources of the transistors.

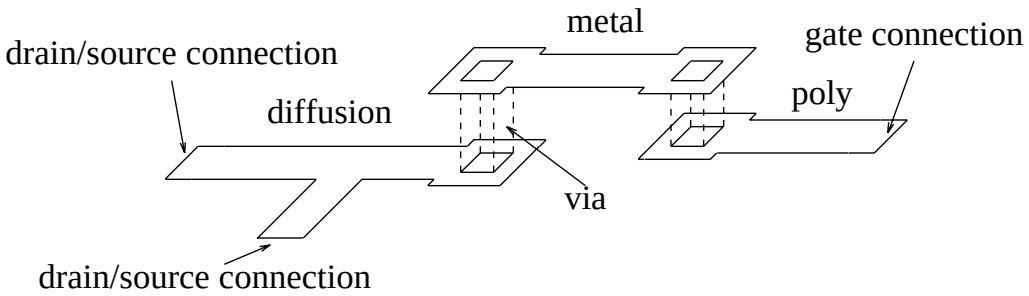


Figure 2.1. Interconnection model for resistance calculation.

In order to compute the resistances, a relation between the terminal potentials and the terminal currents of each interconnection has to be determined. For a separate homogeneous region corresponding to a metal, poly-silicon or diffusion region,

this relation is governed by the Laplace equation, which states that the sum of the partial second-order derivatives of the potential distribution in the *x* direction and in the *y* direction is equal to zero. Using this equation, different methods can be applied to compute the resistances, such as polygonal decomposition, conformal transformation, boundary-element methods, the finite-difference method, and the finite-element method. An overview of these methods is given below. From this overview, we find that the use of the finite-element method for resistance calculation provides several advantages as compared to the other methods that can be used for resistance calculation. These advantages are based on the fact that the finite-element method is generally applicable and that it permits the combination of an acceptable degree of accuracy for complicated interconnection polygons with a reasonable degree of efficiency, even for large interconnections wires. Also, it appears that the finite-element method is easily adapted to the modeling of distributed capacitive effects, as discussed in Section 2.3. Therefore, we will select the finite-element method for the computation of the interconnect resistances. In Section 2.2.2 we will describe the finite-element method in more detail. After that, in Section 2.2.3, we will give some examples in which the finite-element method is used to compute the resistances of some simple - standard shape - interconnection polygons, so as to illustrate the usage of the finite-element method for the resistance calculation of practical VLSI interconnections.

2.2.1.1 Polygonal decomposition

One technique to find the resistances of IC interconnections consists of decomposing each interconnection polygon into a set of basic polygons for which the resistances are easily calculated from the length-to-width ratio of the polygon, or for which they have been obtained using some other method [4, 17, 18]. The resistances of the original polygon are then obtained by combining the resistances of the basic polygons. To decompose the interconnections into their basic polygons, equi-potential lines have to be chosen along which the interconnections are split. Further, an appropriate library of basic polygons together with their resistance values has to be available. Although methods that are based on this technique provide quick and accurate results for simple interconnection wires that have only L-bends and T-crossings etc., one of their disadvantages is that they are not well suited for arbitrarily shaped interconnection polygons. In general, it may be hard or even impossible to find the appropriate equi-potential lines along which the interconnections are split, and the library of basic polygons may be too small to cover the remaining polygons. Therefore the accuracy of these methods can not always be guaranteed.

2.2.1.2 Conformal transformation

Conformal transformation [2] is a mathematical technique to obtain a closed-form expression for the resistance of a conductor polygon by transforming the conductor polygon into another polygon for which the resistance is already known. It is based on the invariance of the Laplacian field under conformal transformation. First, the given conductor pattern is arranged on a complex plane $z = x + iy$. Then the pattern is transformed into a second complex plane $w = u + iy$. The conformal transformation relating w and z is chosen such that the initial pattern is transformed into a pattern for which the resistance is trivial or for which it has previously been computed. Then, the solution for the new pattern is transformed back, using the relation between w and z, to find the resistance of the initial pattern. The method needs human inventivity to make it work and is useful for only a limited set of interconnection polygons. Hence, it is not suited for use in a layout-to-circuit extraction program.

2.2.1.3 Boundary-element methods

Boundary-element methods [10, 11] solve the relation between the currents and voltages of the interconnection by considering the Laplace equation in integral form. In [10] Cauchy's integral formula is used to solve the Laplace equation and in [11] a special application of Green's boundary formula is used to find a solution for the Laplace equation. Both methods subdivide the boundaries of the interconnections into line segments and both methods yield a set of N simultaneous linear equations of N variables (where N is equal or proportional to the number of line segments) from which the relation between the currents and the voltages is computed. Experiments show that these methods provide useful results for small interconnections with a only a few line segments [10, 11] , but the set of equations from which the currents and voltages are solved is non-sparse, which results in inefficient computation times for large interconnection wires with many elements.

2.2.1.4 Finite-difference method

With the finite-difference method [6, 19, 20, 21] the domain of the polygon is divided into a rectangular grid and the potential at each grid point is solved by a discretization of the Laplace equation. This discretization expresses the potential in each grid point as a linear function of the potentials of its four neighboring grid points, and yields a set of N simultaneous linear equations of N variables (where N is the number of grid points) from which the resistances are computed. The set of linear equations is symmetric, positive definite and sparse, and it is efficiently solved for also large problems [21, 22]. However, because of the use of a

rectangular grid, the finite-difference method has some difficulties in modeling non-orthogonal interconnection boundaries.

2.2.1.5 Finite-element method

The finite-element method [3, 5, 7, 8, 9, 23, 24] in its simplest form subdivides the conductor domain into triangles and approximates the potential distribution in the conductor domain by a piece-wise planar function that is defined over these triangles. The minimization of a functional that relates to the consumed energy in the conductor domain is then used as an alternative formulation for the Laplace equation. This is equivalent to modeling each triangle by a delta-type resistor network and solving the final resistances from the relation between the potentials and currents in the total resistor network. Just as with the finite-difference method, the finite-element method yields a sparse, symmetric and positive definite set of N simultaneous linear equations of N variables (where N is equal to the number of nodes of the triangular mesh) from which the resistances are solved. The finite-element method is easily adopted for conductors regions that have sub-domains with different sheet conductivities, and - because of the use of triangular mesh - the method is also suited for interconnections that have non-orthogonal boundaries.

2.2.2 The finite-element method

In this section, we will give an extensive description of the finite-element method for resistance calculation. The description that is given is, among other things, based on previous descriptions that have been given in [8, 23] and [25]. First, we give a precise definition of the resistance calculation problem. Next, we describe a strategy to obtain an approximate solution based on the finite-element theory where we will use linear (first-order) shape functions to approximate the potential distribution on the conductors. Then, we will give the method to actually obtain the solution. Finally, we show an analogy between the finite-element method and the solution of a resistance mesh network.

2.2.2.1 Problem definition

The conductor is represented by a domain Ω that has a sheet conductivity σ, a boundary Γ, and ideally conducting electrodes at intervals on the boundary $\gamma_1 \cdots \gamma_M$, representing the terminals of the interconnection (see Figure 2.2). The domain Ω is further subdivided into different homogeneous regions Ω_1, $\Omega_2 \cdots$, each having a constant sheet conductivity σ_1, $\sigma_2 \cdots$ and representing the different types of interconnection layers and/or vias between them. When E is the electric field at a point (x, y) inside the domain Ω, the current density j at that point

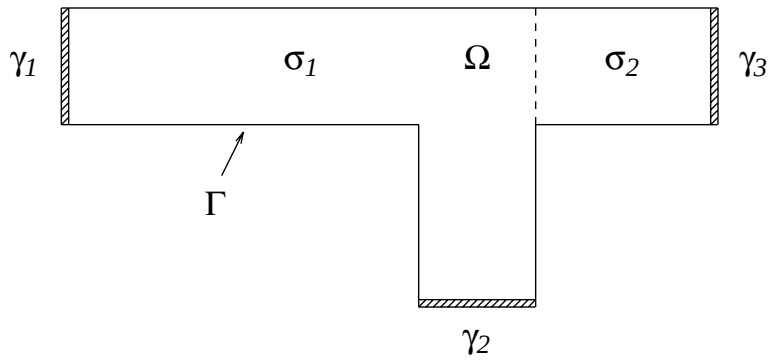


Figure 2.2. An interconnection Ω with boundary Γ and terminals γ_1, γ_2 and γ_3.

is given by

$$j = \sigma E \tag{2.1}$$

according to Ohm's law. Since we are considering a quasi-stationary state, no charge is accumulated, and assuming no injection of charge we find

$$\nabla \cdot j = 0 \tag{2.2}$$

where $\nabla = \{\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\}$. Using the relation

$$E = -\nabla \phi \tag{2.3}$$

we then find from 2.1 and 2.2 for the potential distribution ϕ

$$\nabla \cdot (\sigma \nabla)\phi = 0. \tag{2.4}$$

For a separate homogeneous region (where σ is constant) this results in

$$\nabla^2 \phi = 0, \tag{2.5}$$

where $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$, which is known as the Laplace equation.

For resistance calculation, the purpose is to determine a relation between the terminal potentials $\Phi_1 \cdots \Phi_M$ and the currents $I_1 \cdots I_M$ flowing into the terminals resulting in the matrix relation

$$\begin{bmatrix} I_1 \\ I_2 \\ . \\ I_M \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} & . & Y_{1M} \\ Y_{21} & Y_{22} & . & Y_{2M} \\ . & . & . & . \\ Y_{M1} & Y_{M2} & . & Y_{MM} \end{bmatrix} \begin{bmatrix} \Phi_1 \\ \Phi_2 \\ . \\ \Phi_M \end{bmatrix}. \tag{2.6}$$

Hence, at the terminal positions γ_i we have for Equation 2.4 the Dirichlet boundary condition

$$\phi = \Phi_i. \tag{2.7}$$

Since no current is flowing through Γ other than at the terminals positions, we have for the remaining parts of Γ the Neumann condition

$$\frac{\partial \phi}{\partial n} = 0, \tag{2.8}$$

where n is the outward normal on Γ.

In addition, the currents I_i ($i = 1 \cdots M$) flowing into the terminals are given by

$$I_i = \int_{\gamma_i} \sigma \frac{\partial \phi}{\partial n} dl. \tag{2.9}$$

2.2.2.2 Solution strategy

To find a solution for 2.4 with boundary conditions 2.7 and 2.8 we will make use of the following property:

Theorem 2.1 :

Consider a conductor that has a domain Ω, a sheet conductivity σ and a boundary Γ. The boundary Γ is divided up into two sets of boundary parts Γ^1 and Γ^2 such that on Γ^1 we have the Dirichlet condition

$$\phi = \alpha, \tag{2.10}$$

where α is a potential function along Γ^1, and on Γ^2 we have the Neumann condition

$$\frac{\partial \phi}{\partial n} = \frac{\beta}{\sigma}, \tag{2.11}$$

where β is a current density function along Γ^2. Then, the solution for the potential distribution ϕ in the domain Ω that satisfies

$$\nabla \cdot (\sigma \nabla) \phi = 0 \tag{2.12}$$

and the boundary conditions 2.10 and 2.11 is given by the potential distribution

ϕ that minimizes the functional

$$\chi(\phi) = \int_{\Omega} \frac{\sigma}{2} |\nabla\phi|^2 \, dS - \int_{\Gamma^2} \phi \, \beta \, dl. \tag{2.13}$$

Proof:

Assume the exact solution is given by ϕ. Now, consider a perturbation function ψ that is such that $\psi = 0$ on Γ^1, and $\frac{\partial\psi}{\partial n} = 0$ on Γ^2. Hence, the function $\phi + \psi$ satisfies the boundary conditions.

The value of χ for $\phi + \psi$ is given by

$$\begin{aligned}
\chi(\phi+\psi) &= \int_{\Omega} \frac{\sigma}{2} |\nabla(\phi+\psi)|^2 \, dS - \int_{\Gamma^2} (\phi+\psi) \, \beta \, dl \\
&= \int_{\Omega} \frac{\sigma}{2} |\nabla\phi|^2 \, dS + \int_{\Omega} \frac{\sigma}{2} |\nabla\psi|^2 \, dS \\
&\quad + \int_{\Omega} \sigma \, \nabla\phi\cdot\nabla\psi \, dS - \int_{\Gamma^2} \phi \, \beta \, dl - \int_{\Gamma^2} \psi \, \beta \, dl. \\
&= \chi(\phi) + \int_{\Omega} \frac{\sigma}{2} |\nabla\psi|^2 \, dS \\
&\quad + \int_{\Omega} \sigma \, \nabla\phi\cdot\nabla\psi \, dS - \int_{\Gamma^2} \psi \, \beta \, dl.
\end{aligned} \tag{2.14}$$

Using Green's first identity

$$\int_{\Omega} \psi \, \nabla\cdot(\sigma\nabla\phi) \, dS + \int_{\Omega} \sigma\nabla\phi \cdot \nabla\psi \, dS = \int_{\Gamma} \psi \, \sigma \frac{\partial\phi}{\partial n} \, dl, \tag{2.15}$$

expression 2.14 may be rewritten as

$$\begin{aligned}
\chi(\phi+\psi) &= \chi(\phi) + \int_{\Omega} \frac{\sigma}{2} |\nabla\psi|^2 \, dS - \int_{\Omega} \psi \, \nabla\cdot(\sigma\nabla\phi) \, dS \\
&\quad + \int_{\Gamma^1} \psi \, \sigma \frac{\partial\phi}{\partial n} \, dl + \int_{\Gamma^2} \psi \, (\sigma \frac{\partial\phi}{\partial n} - \beta) \, dl.
\end{aligned} \tag{2.16}$$

The third term equals zero since ϕ satisfies 2.12, the fourth term equals zero since $\psi = 0$ on Γ^1, and the fifth term equals zero since $\beta = \sigma \frac{\partial\phi}{\partial n}$ on Γ^2. Hence, we find

$$\chi(\phi+\psi) = \chi(\phi) + \int_{\Omega} \frac{\sigma}{2} |\nabla\psi|^2 \, dS$$

$$\geq \chi(\phi), \tag{2.17}$$

for each admissible function ψ. Thus the exact solution ϕ provides a minimum for the functional as defined by 2.13.

□

Note:

For problems where the boundary conditions are given by fixed potentials and/or zero currents along the boundary ($\Gamma^2 = \varnothing$ or $\beta \equiv 0$) the expression for χ as given in 2.13 relates to the dissipated energy in the conductor. The energy that is dissipated per second in the domain Ω of the conductor is equal to

$$W(\phi) = \int_{\Omega} j \cdot E \, dS = \int_{\Omega} \sigma \, |\nabla\phi|^2 \, dS. \tag{2.18}$$

Expression 2.18 is (apart from a factor 2) equivalent to 2.13 if $\Gamma^2 = \varnothing$ or $\beta \equiv 0$. Thus, in Figure 2.2, for situations where only the potentials at the terminals of the conductor are specified, the searching for a potential distribution ϕ that satisfies 2.4 (or the Laplace equation 2.5 if σ is constant) is equivalent to searching for a potential distribution ϕ that provides a minimum for the energy that is dissipated in the conductor.

The solution strategy is now as follows. To find the solution for 2.4 with boundary conditions 2.7 and 2.8 we approximate the potential distribution ϕ in the domain Ω by a piece-wise planar function. This is achieved by subdividing the region Ω into triangles (see Figure 2.3a) and approximating the potential distribution ϕ^e on each element e with nodes i, j and k by a linear function of the form

$$\phi^e(x, y) = N_i(x, y)\,\phi_i + N_j(x, y)\,\phi_j + N_k(x, y)\,\phi_k, \tag{2.19}$$

where ϕ_l ($l = i, j, k$) is the potential at node l, and N_l ($l = i, j, k$) is a shape function that is 1 at node l and that linearly decreases towards zero from node l to the opposite edge of node l (see Figure 2.3b). For the sake of convenience, we construct the triangles such that each triangle is fully part of one homogeneous region, so that σ is constant within one triangle.

The potential distribution over Ω is made continuous by sharing each node of a triangle between all other triangles that are adjacent to the node. The total potential distribution ϕ in the domain Ω is then given by

$$\phi(x, y) = \sum_{i=1}^{N} N_i(x, y)\, \phi_i, \tag{2.20}$$

where $1 \cdots N$ is the total set of nodes of the finite-element mesh.

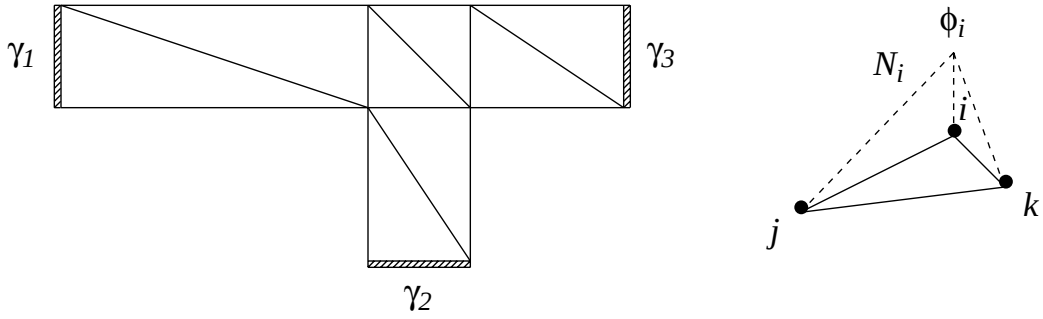


Figure 2.3. (a) Subdividing an interconnection into triangles. (b) A single triangle with nodes i, j and k and its shape function N_i.

Following Theorem 2.1, we search for a set of node potentials $\{ \phi_i \mid i = 1 \cdots N \}$ that minimizes the functional χ for a certain (unknown) admissible current density β along the entire boundary Γ. The function β is admissible if β is constant along one edge of a triangle (this follows from the potential distribution as given in 2.19) and $\int_{\Gamma} \beta\, dl = 0$ (because of the current balance). From function theory, it follows that the minimum for χ will be reached when

$$\frac{\partial \chi(\phi)}{\partial \phi_i} = 0 \qquad (i = 1 \cdots N). \tag{2.21}$$

Then, the currents through the edges of the triangles are associated with the nodes that are connected to the edges. This will provide a set of N simultaneous equations of N variables, relating the potentials of all nodes to the currents that are flowing into the nodes.

Next, we account for the boundary conditions by requiring that the potentials of the nodes that are part of the same terminal are equal to the potential of the terminal, and by requiring that the sum of the currents of the nodes that are part of the same terminal is equal to the current for that terminal. For the nodes that are not part of a terminal, the currents are zero. When n_i is the number of nodes belonging to terminal i this will provide a set of $N - \sum_{i=1}^{M} (n_i - 1)$ simultaneous equations, relating the potentials of the non-terminal (i.e. internal) nodes and the potentials of the terminals to the currents of the terminals. From this last set of

equations the potentials of the internal nodes are eliminated, and in this way yields a relation between the terminal potentials and the terminal currents similar to relation 2.6.

In order to minimize χ for the total domain Ω, we will first derive a set of equations that is used to minimize χ for a single element. Then, using the result for a single element, we derive a set of equations that is valid for the total finite-element mesh. Finally, we solve the relation between the terminal potentials and the terminal currents.

2.2.2.3 Solution method

Based on the definition of the shape functions N_i, N_j and N_k, an alternative expression for the total potential distribution ϕ^e on a triangle is given by

$$\phi^e(x, y) = \alpha_1 + \alpha_2 x + \alpha_3 y = [\ 1\ \ x\ \ y\ ] \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}. \tag{2.22}$$

Expressing the node potentials ϕ_i, ϕ_j and ϕ_k according to 2.22 yields

$$\begin{bmatrix} \phi_i \\ \phi_j \\ \phi_k \end{bmatrix} = \begin{bmatrix} 1 & x_i & y_i \\ 1 & x_j & y_j \\ 1 & x_k & y_k \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}, \tag{2.23}$$

where (x_i, y_i), (x_j, y_j) and (x_k, y_k) are coordinates of nodes i, j and k.

Upon solving the coefficients α_1, α_2 and α_3 from 2.23 and inserting the result into 2.22 we obtain

$$\phi^e(x, y) = [\ 1\ \ x\ \ y\ ] \begin{bmatrix} 1 & x_i & y_i \\ 1 & x_j & y_j \\ 1 & x_k & y_k \end{bmatrix}^{-1} \begin{bmatrix} \phi_i \\ \phi_j \\ \phi_k \end{bmatrix}. \tag{2.24}$$

By comparison with 2.19, we then find for the shape functions

$$\begin{bmatrix} N_i(x, y) \\ N_j(x, y) \\ N_k(x, y) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_i & x_j & x_k \\ y_i & y_j & y_k \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ x \\ y \end{bmatrix}$$

$$= \frac{1}{2\Delta^e} \begin{bmatrix} x_j y_k - x_k y_j & y_j - y_k & x_k - x_j \\ x_k y_i - x_i y_k & y_k - y_i & x_i - x_k \\ x_i y_j - x_j y_i & y_i - y_j & x_j - x_i \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \end{bmatrix}, \tag{2.25}$$

where

$$\Delta^e = \frac{(x_j y_k - x_k y_j) + (x_k y_i - x_i y_k) + (x_i y_j - x_j y_i)}{2}, \tag{2.26}$$

is the area of the element.

For a single element that has a boundary Γ^e and a current density β that is defined along the entire boundary, the functional 2.13 is written as

$$\chi^e(\phi) = \int_{\Delta^e} \frac{\sigma}{2} \left\{ \left(\frac{\partial \phi^e}{\partial x} \right)^2 + \left(\frac{\partial \phi^e}{\partial y} \right)^2 \right\} dx\, dy \quad - \quad \int_{\Gamma^e} \phi^e \, \beta \, dl. \tag{2.27}$$

This functional has an extremum, i.e. minimum, for

$$\frac{\partial \chi^e}{\partial \phi_i} = 0, \quad \frac{\partial \chi^e}{\partial \phi_j} = 0 \text{ and } \frac{\partial \chi^e}{\partial \phi_k} = 0. \tag{2.28}$$

When inserting 2.25 into 2.19 and putting the result into 2.27 we derive from 2.28

$$\begin{bmatrix} Y_{ii}^e & Y_{ij}^e & Y_{ik}^e \\ Y_{ji}^e & Y_{jj}^e & Y_{jk}^e \\ Y_{ki}^e & Y_{kj}^e & Y_{kk}^e \end{bmatrix} \begin{bmatrix} \phi_i^e \\ \phi_j^e \\ \phi_k^e \end{bmatrix} = \begin{bmatrix} i_i^e \\ i_j^e \\ i_k^e \end{bmatrix}, \tag{2.29a}$$

where

$$Y_{ij}^e = \int_{\Delta^e} \sigma \left(\frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) dx\, dy \tag{2.29b}$$

$$i_i^e = \int_{\Gamma^e} \beta \, N_i \, dl. \tag{2.29c}$$

Evaluation of the integrals leads to

$$\begin{bmatrix} G_{ij} + G_{ik} & -G_{ij} & -G_{ik} \\ -G_{ij} & G_{ij} + G_{jk} & -G_{jk} \\ -G_{ik} & -G_{jk} & G_{ij} + G_{jk} \end{bmatrix} \begin{bmatrix} \phi_i^e \\ \phi_j^e \\ \phi_k^e \end{bmatrix} = \begin{bmatrix} i_i^e \\ i_j^e \\ i_k^e \end{bmatrix}, \tag{2.30}$$

where

$$G_{ij} = \frac{\sigma}{4\Delta^e} [\ (y_k - y_j)(y_k - y_i) + (x_j - x_k)(x_i - x_k)\ ]$$

$$G_{ik} = \frac{\sigma}{4\Delta^e} [\ (y_j - y_k)(y_j - y_i) + (x_k - x_j)(x_i - x_j)\ ] \qquad (2.31)$$

$$G_{jk} = \frac{\sigma}{4\Delta^e} [\ (y_i - y_k)(y_i - y_j) + (x_k - x_i)(x_j - x_i)\ ].$$

Thus we have obtained the set of equations that minimizes χ for a single element. To find the extremum of χ for the total system we note that when

$$\int_{\zeta} \phi\ \beta\ dl = 0, \qquad (2.32)$$

where ζ is the set of triangle edges that are not part of the boundary Γ, χ is obtained by piece-wise integration over the finite elements from

$$\chi = \sum_e \chi^e, \qquad (2.33)$$

where *e* ranges over all triangles. Hence when assuming 2.32, a solution to 2.21 is obtained from

$$\sum_e \frac{\partial \chi^e}{\partial \phi_i} = 0 \qquad (i = 1\ \cdots\ N), \qquad (2.34)$$

where *e* ranges over all triangles.

Writing out 2.34 yields

$$\begin{bmatrix} Y_{11} & Y_{12} & . & Y_{1N} \\ Y_{21} & Y_{22} & . & Y_{2N} \\ . & . & . & . \\ Y_{N1} & Y_{N2} & . & Y_{NN} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ . \\ \phi_N \end{bmatrix} = \begin{bmatrix} i_1 \\ i_2 \\ . \\ i_N \end{bmatrix}, \qquad (2.35a)$$

where

$$Y_{ij} = \sum_e Y_{ij}^e \qquad (2.35b)$$

$$i_i = \sum_e i_i^e. \qquad (2.35c)$$

In order to satisfy 2.32, we use for every node *i* that is on Γ

$$i_i = \tfrac{1}{2}\ l_a\ \beta_a + \tfrac{1}{2}\ l_b\ \beta_b \qquad (i = 1\ \cdots\ N \wedge i\ on\ \Gamma), \qquad (2.36)$$

where l_a and l_b are the lengths of the two boundary edges that are connected to

node i and β_a and β_b are the current densities through these edges (see Figure 2.4), and for the remaining nodes

$$i_i = 0 \qquad (i = 1 \cdots N \wedge i \text{ not on } \Gamma). \tag{2.37}$$

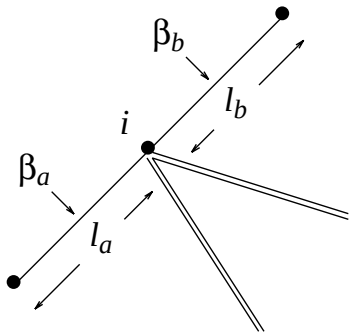


Figure 2.4. A boundary node i with boundary edges a and b.

Then (recall that β is constant along one edge and remark that $\int N_i \, dl = \frac{1}{2}$ for one edge that is connected to i), by insertion of 2.29c and 2.35c into 2.36 and 2.37, it follows that

$$\int_{\eta_i} \beta \, N_i \, dl = 0 \qquad (i = 1 \cdots N) \tag{2.38}$$

for η_i ranging over the triangles edges connected to node i that are not part of the boundary Γ, which implies

$$\int_{\eta_i} \phi_i \, \beta \, N_i \, dl = 0 \qquad (i = 1 \cdots N), \tag{2.39}$$

or

$$\sum_{i=1}^{N} \int_{\eta_i} \phi_i \, \beta \, N_i \, dl = 0, \tag{2.40}$$

which is equivalent to 2.32.

At this point we have obtained a set of simultaneous equations (2.35) relating the potentials of the nodes of the finite-element mesh to the the currents that are flowing through the boundary Γ. For every node i that is not on a terminal it follows from 2.36 and 2.37

$$i_i = 0 \qquad (i = 1 \cdots N \wedge i \notin \{ \gamma_k \mid k = 1 \cdots M \}). \tag{2.41}$$

For the remaining nodes we have

$$\sum_{i \in \gamma_k} i_i = I_k \qquad (k = 1 \ \cdots \ M). \tag{2.42}$$

And for the potentials of the nodes that are part of a terminal it must hold that

$$\phi_i = \Phi_k \qquad (i \in \{ \gamma_k \mid k = 1 \ \cdots \ M \}). \tag{2.43}$$

Now, let the nodes $1 \ \cdots \ N$ be numbered such that $1 \ \cdots \ L$ are the nodes of the terminals and $L + 1 \ \cdots \ N$ are the other nodes of the finite-element mesh ($L = \sum_{i=1}^{M} n_i$). Then Equation 2.35 may be written as

$$\begin{bmatrix} i_t \\ i_n \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} \begin{bmatrix} \phi_t \\ \phi_n \end{bmatrix}, \tag{2.44}$$

where Y_{11} is a $L \times L$ submatrix from Y, $Y_{12} = Y_{21}^T$ is a $L \times (N-L)$ submatrix from Y, Y_{22} is a $(N-L) \times (N-L)$ submatrix from Y, i_t and ϕ_t are respectively a current vector and a potential vector of size L for the terminal nodes, and i_n and ϕ_n are respectively a current vector and a potential vector of size $N-L$ for the internal nodes.

From 2.41 it follows

$$i_n = 0, \tag{2.45}$$

where 0 is a vector containing $N - L$ zeros.

Further let F be an $N \times M$ incidence matrix of nodes and terminals in which

$$F_{ij} = \begin{cases} 1 & \text{if node } i \text{ is on terminal } j \\ 0 & \text{otherwise.} \end{cases} \tag{2.46}$$

Then from 2.42 we find

$$i_t = F^{-T} I \tag{2.47}$$

and from 2.43 we have

$$\phi_t = F \, \Phi. \tag{2.48}$$

Substitution of 2.45, 2.47 and 2.48 in 2.44 yields

$$\begin{bmatrix} I \\ 0 \end{bmatrix} = \begin{bmatrix} F^T Y_{11} F & F^T Y_{12} \\ Y_{21} F & Y_{22} \end{bmatrix} \begin{bmatrix} \Phi \\ \phi_n \end{bmatrix}, \tag{2.49}$$

which is a set of equations that relates the potentials of the internal nodes and the potentials of the terminals to the currents of the terminals. From this set of

equations, the potentials of the internal nodes ϕ_n can be eliminated since Y_{22} is strictly diagonal dominant, yielding

$$I = (F^T\ Y_{11}\ F - F^T\ Y_{12}\ Y_{22}^{-1}\ Y_{21}\ F)\ \Phi, \tag{2.50}$$

which is the desired set of equations, see 2.6.

The transformation of the admittance matrix Y in 2.49 into the final admittance matrix Y in 2.50 is, for example, executed by the Gaussian elimination of rows and columns $M+1\ \cdots\ N-L+M$ of Y in 2.49 [26]. When k is the current number of rows/columns of Y and Y_{ij} is the entry of Y that is in row i and column j, then the elimination of row/column k, yielding a new matrix $Y^{(2)}$ with entries $Y_{ij}^{(2)}$ ($i = 1\ \cdots\ k-1$, $j = 1\ \cdots\ k-1$), is performed by

$$\begin{aligned} &\text{for } i := 1\ \cdots\ k \\ &\quad \text{for } j := 1\ \cdots\ k \\ &\qquad Y_{ij}^{(2)} := Y_{ij} - \frac{Y_{kj}\ Y_{ik}}{Y_{kk}}. \end{aligned} \tag{2.51}$$

Because of 2.30, Y is symmetrical and positive definite. Further, the original matrix Y in 2.44 is sparse since each node in the finite-element mesh has only a few edges connected to it, independent of the size of mesh.

2.2.2.4 Resistor network representation

The finite-element method for resistance calculation is equivalent to the solution of a resistor network when replacing each triangle of the finite-element mesh by a delta-type resistor network as shown in Figure 2.5, where the conductance G_{ij} of each edge *(i, j)* of the triangle is given by G_{ij} in 2.31. The analogy follows by comparing the admittance matrix of the resistor network for a triangle with Equation 2.30, and by comparing the admittance matrix of the total resistor mesh for the total finite-element mesh with 2.35. From these comparisons it is found that the relation between the potentials and the currents in the resistor network for a triangle is identical to 2.30, and that the relation between the potentials and the currents in the total resistor mesh is identical to 2.35. Thus, the solution of the finite-element mesh, i.e. the final resistances of the interconnection, is alternatively obtained from the relation between the terminal potentials and the terminal currents in the the total resistor mesh.

The transformation of Equation 2.35 or 2.44 into Equation 2.50 is equivalent in the network representation to the collapsing of all nodes that belong to the same terminal. An elimination of a row and column in Y corresponds, in the network representation, to the elimination of an internal node or, what is otherwise called, a

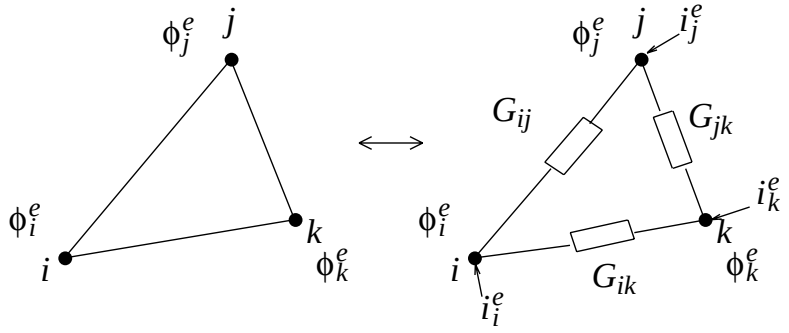


Figure 2.5. A triangle and its resistor network equivalence.

star-triangle transformation for that node. The elimination of a node is illustrated in Figure 2.6. When eliminating node k, the new conductance $G_{ij}^{(2)}$ between a node pair i, j is found from (compare with 2.51)

$$G_{ij}^{(2)} := G_{ij} + \frac{G_{kj}\, G_{ik}}{\sum\limits_{i=1,\, i\neq k}^{N} G_{ki}}, \tag{2.52}$$

where $\sum\limits_{i=1,\, i\neq k}^{N} G_{ki}$ is the sum of the conductances connected to node k.

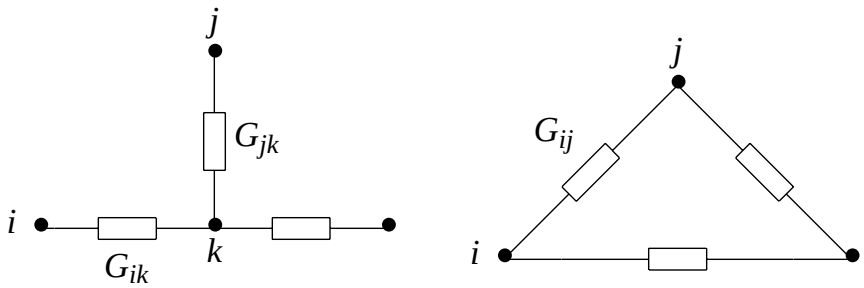


Figure 2.6. The network equivalence of an elimination of a row/column k in Y''.

As an example of a resistor network representation for a finite-element mesh, we consider the finite-element mesh in Figure 2.3. The equivalent resistor mesh when $\sigma = 1$ S/□ is given in Figure 2.7. The conductances of edges opposite to a right-angled corner are zero since it follows from 2.31 that $G_{ij} = 0$ when the inner product of the edges (i, k) and (j, k) is zero. Therefore, in general, each pair of abutting triangles that together form a rectangle may be substituted by a 4-port resistor network as shown in Figure 2.8, where the conductances of the 4-port resistor network are given by the corresponding conductances of the edges of the triangles.

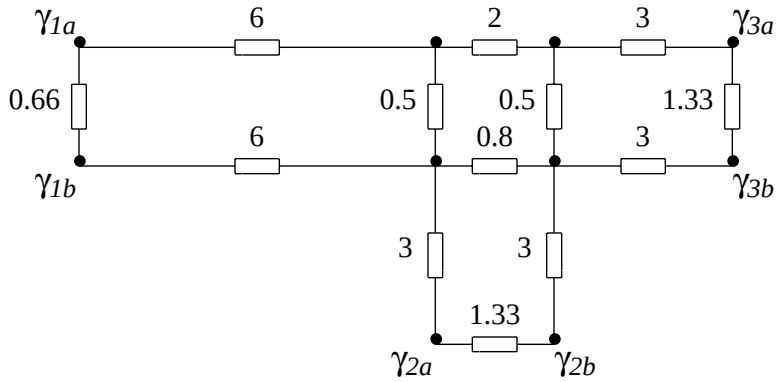


Figure 2.7. Resistor network representation for the finite-element mesh in Figure 2.3.

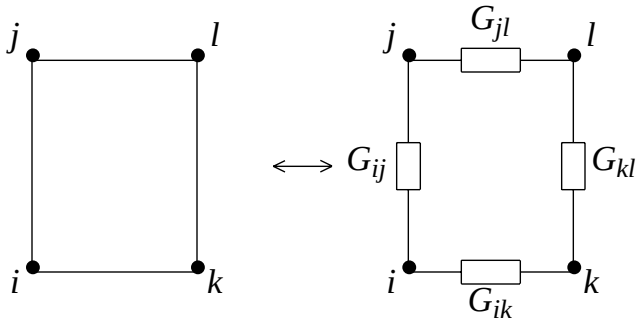


Figure 2.8. A rectangle and its resistor network representation.

Finally, Figure 2.9 shows the network that is obtained from Figure 2.7 after collapsing the terminal nodes and eliminating the internal nodes of the mesh. This network has only nodes that correspond to a terminal of the interconnection and it has a resistor between each pair of terminals.

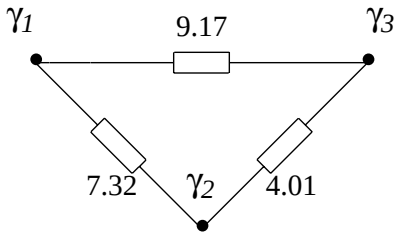


Figure 2.9. Terminal resistances obtained from the resistor mesh shown in Figure 2.7.

2.2.3 Examples

A few examples are given that illustrate the usage of the finite-element method for the resistance calculation of interconnection polygons occurring in VLSI circuits. The interconnect polygons for which the resistances are calculated are shown in Figure 2.10a-c. For all interconnection polygons the sheet conductivity σ is assumed to be *1 S/□*. The resistance of each polygon is calculated for different complexities of the finite-element mesh. The results are shown in Table 2.1, 2.2 and 2.3 respectively, where the complexity of the finite-element mesh is indicated by the number of nodes of the finite-element mesh. To obtain information about accuracy, the results have been compared with values that are presented in literature.

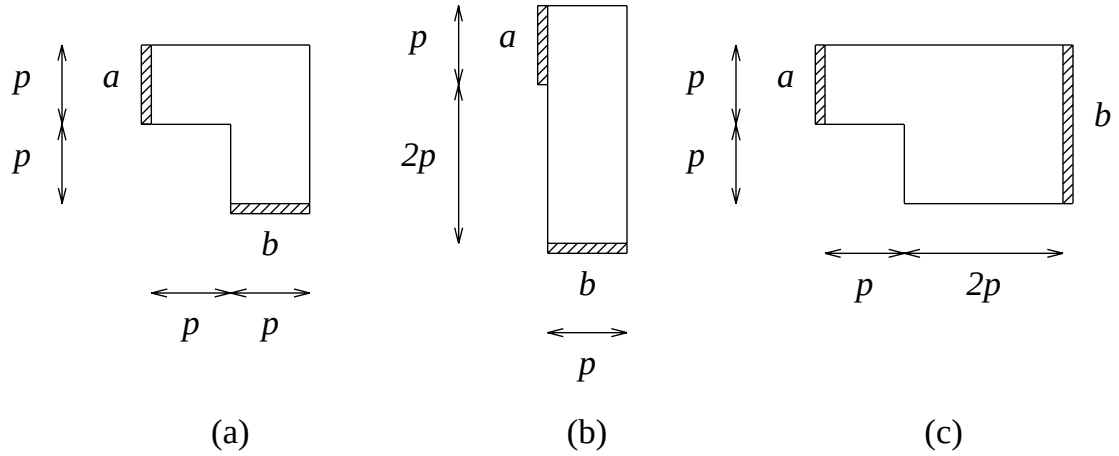


Figure 2.10. Interconnection polygon examples.

Table 2.1. Resistances for Figure 2.10a.

nodes	resistance in Ω	error in %
7	2.167	15
8	2.286	11
15	2.388	6.7
37	2.470	3.4
106	2.540	0.74
304	2.550	0.35
literature [2]: 2.559 Ω (upper bound)		

Some of the finite-element meshes that have been used to obtain the results in Table 2.1 are shown in Figure 2.11a-d. In Figure 2.11d a relatively high degree of refinement of the finite-element mesh is used for positions that are near the inside

Table 2.2. Resistances for Figure 2.10b.

nodes	resistance in Ω	error in %
5	1.778	28
6	2.133	14
16	2.294	7.1
36	2.350	4.8
149	2.406	2.6
375	2.455	0.47
literature [2]: 2.469 Ω (upper bound)		

Table 2.3. Resistances for Figure 2.10c.

nodes	resistance in Ω	error in %
7	2.082	7.4
8	2.082	7.4
14	2.126	5.5
41	2.199	2.3
136	2.236	0.6
342	2.245	0.2
literature [4]: 2.25 Ω		

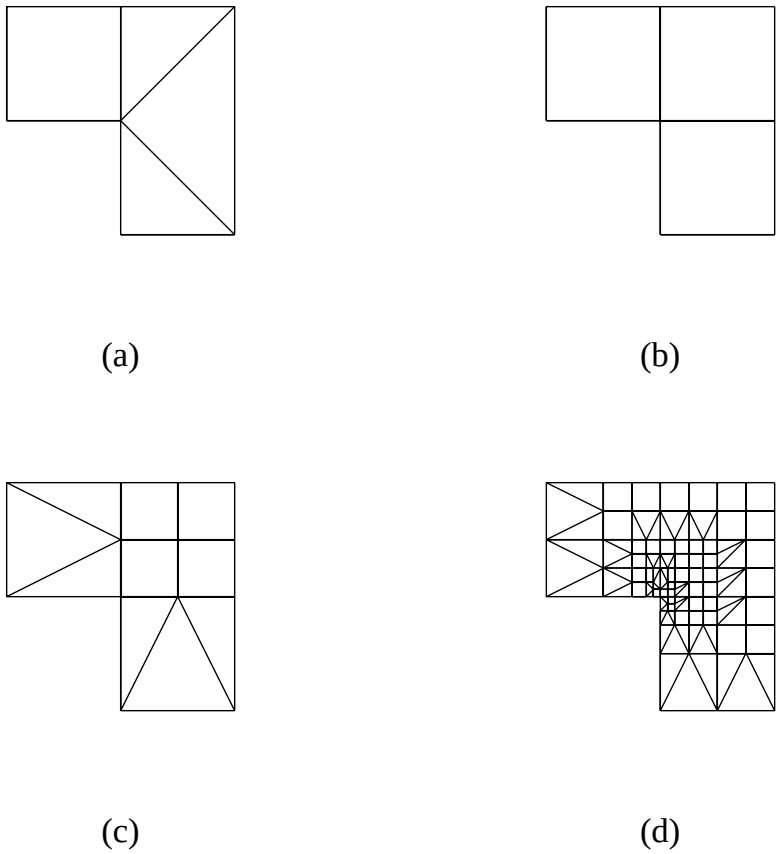
of the corner of the interconnect polygon, so as to optimally model the non-uniformity of the potential distribution in the interconnection.

From Tables 2.1, 2.2 and 2.3 we conclude that useful results (e.g. accuracy better than 10-15 %) have already been obtained for relatively low complexities of the finite-element mesh. In fact, a simple subdivision as in Figure 2.11b might be sufficient to calculate the resistances of VLSI interconnections.

2.3 Capacitance distribution

2.3.1 Introduction

When neglecting the inductive effects, the transmission behavior of an IC interconnection is determined by distributed resistance and distributed capacitance. As an example we consider a one-dimensional interconnection line that has, at a position x, a resistance per unit length $r(x)$ and a capacitance per unit length $c(x)$ (see Figure 2.12). The potential $\phi(x, t)$ and the current $i(x, t)$ in positive x direction, at position x and time t, are found from



(a) (b)

(c) (d)

Figure 2.11. Finite-element meshes for the interconnection in Figure 2.10a; (a) number of nodes is 7; (b) number of nodes is 8; (c) number of nodes is 15; (d) number of nodes is 106.

$$\frac{\partial \phi(x,\, t)}{\partial x} \;=\; -\, i(x,\, t)\, r(x), \tag{2.53}$$

$$\frac{\partial i(x,\, t)}{\partial x} \;=\; -\, c(x)\, \frac{\partial \phi(x,\, t)}{\partial t}. \tag{2.54}$$

Upon elimination of the current $i(x, t)$ we find for the propagation of voltage changes along the interconnection line

$$\frac{\partial^2 \phi(x,\, t)}{\partial x^2} \;-\; \frac{1}{r(x)}\, \frac{\partial r(x)}{\partial x}\, \frac{\partial \phi(x)}{\partial x} \;=\; r(x)\, c(x)\, \frac{\partial \phi(x,\, t)}{\partial t}. \tag{2.55}$$

Experiments show that the distributed RC effects of VLSI interconnections are accurately modeled (see e.g. [12, 13]) by subdividing the interconnections into pieces and replacing each piece by a lumped RC section that is either an L-section, a π-section or a T-section (see Figure 2.13). For each piece, the total capacitance

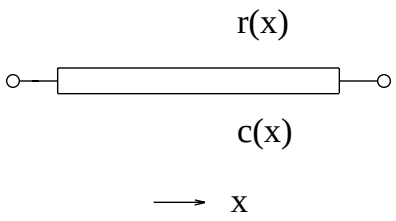


Figure 2.12. A one-dimensional distributed RC line.

C of the piece is thereby divided over the nodes of the lumped resistance model that has, as does the interconnection piece, a total resistance *R*. In order to obtain sufficiently accurate results, the interconnection should be subdivided into a sufficient number of pieces, such that the distributed properties of the interconnection are accurately reflected in the RC ladder network.

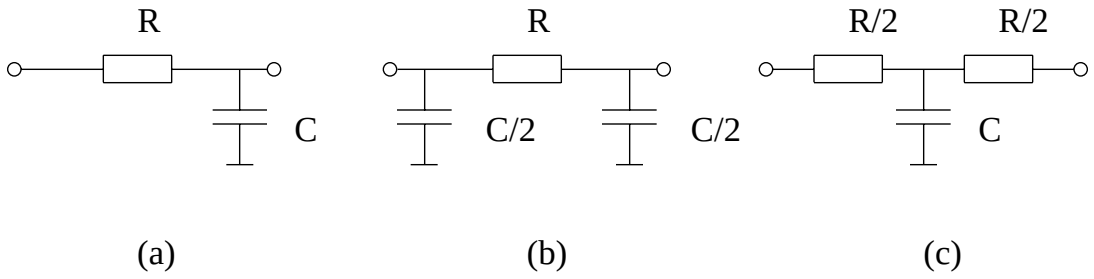


Figure 2.13. (a) L-section, (b) π-section, (c) T-section.

In practice, for arbitrarily IC interconnections, however, a few problems will arise with this method: (1) It may be hard to find interconnection pieces that are accurately represented by an L, π or T section because of irregular interconnection polygons and non-uniformly distributed resistive and capacitive effects and/or (2) the number of sections that are required to obtain an accurate RC model for the interconnection may be too many to allow an efficient verification of the behavior of the circuit afterwards. To solve these problems, some extensions to the RC modeling method have been described [14, 15, 16]. All of these methods use a technique in which, first, a complex RC network is constructed that models the distributed resistive and capacitive effects of the interconnection in detail. Then a network reduction technique is applied to the initial network to obtain a final network that has a much lower number of nodes and elements but that displays approximately the same transmission behavior as the initial network. To guarantee a close resemblance between the transmission behavior of the initial network and the transmission behavior of the final network when transforming the initial network into the final network, all methods preserve the value of the Elmore time constant [27] between the terminals of the network.

In [14] a method is described that first models each interconnection through a detailed RC tree, and then derives an L-section for each output node in the RC tree. To guarantee that the transmission behavior of each L-section closely matches the transmission behavior of the original network, the method assures that the value of the Elmore time constant between the input node and an output node in the final network is equal to its corresponding value in the original network. A disadvantage of this method is that it assumes a certain direction of signal flow in the network and that it is not suited for networks containing resistance loops. Moreover, the resistances that are found in the final network are in general not equal to the total resistances between the corresponding terminal pairs in the initial network.

In [15, 28] a method is described that uses a finite-element mesh to construct the initial RC network. The initial RC network is then transformed into a final network that has as nodes only the nodes that correspond to the terminals of the interconnection. The Elmore time constants between the terminals of the interconnection in all directions are thereby unchanged with respect to their value in the initial network. In addition, the resistances in the final network are computed such that the relation between the static voltages and the static currents of the terminals in the initial network is preserved in the final network. To achieve the desired transformation, a matrix simplification technique is utilized that neglects the higher-order terms of the Laplace variable s in the node admittance matrix of the final network. A drawback of this method is that its matrix simplification technique introduces coupling capacitances in the final model.

The method as described in [16] strongly resembles the method described in [15, 28] but uses a different network reduction technique to transform the initial RC mesh into the final RC model. The node reduction technique that is used here preserves the Elmore time constants of the initial network in the final network without introducing any extra coupling capacitances. Moreover, the method is also suited to model the coupling capacitances between the interconnections.

Because it is more general than the method described in [14] and because the network reduction technique described in [16] has an improved network reduction technique, in comparison with [15, 28], we will use the method of [16] to find a distribution of the interconnect capacitances over the lumped resistance model of the interconnection. We will first discuss the construction of an RC mesh from a VLSI interconnection to model the distributed RC effects of the interconnection in detail (Section 2.3.2). Then we will give the definition of the Elmore time constant (Section 2.3.3). Next we will describe the algorithm that is used in [16] to reduce a complex RC network into a simple RC network while preserving the

Elmore time constants between the terminals (Section 2.3.4). Finally we will give some examples that illustrate the accuracy of the RC network reduction technique (Section 2.3.5).

2.3.2 RC mesh construction

The initial RC mesh that is constructed for an interconnection is used to provide a first discretization of the distributed RC effects of the interconnection by means of simple, uniform, lumped RC sections. To construct the initial RC mesh, the same finite-element mesh may be used as is used for resistance calculation. However, at positions that correspond to regions in which the capacitance is non-uniformly distributed over the elements, the finite-element mesh should further be refined. Each element of the finite-element mesh should be such that it has an approximately uniformly distributed capacitance over its surface and, if edge capacitances are specified, an approximately uniformly distributed edge capacitance over its edges. Then, after replacing each triangle of the finite-element mesh by a 3-port RC section, and after replacing each rectangle of the finite-element mesh by a 4-port RC section, which are extensions of the equivalent resistance networks for the triangle and rectangle derived in Section 2.2, a lumped RC mesh is obtained that accurately represents the distributed RC effects of the interconnection. In the following, we will discuss the derivation of such a lumped RC section for each element of the finite-element mesh. We will subsequently describe a method for the discretization of the distributed ground capacitances of elements and a method for the discretization of the distributed coupling capacitances between the elements. The correctness of the method to assign the ground capacitances is, to a certain extent, verified in Section 2.3.5.

2.3.2.1 Ground capacitances

For a triangle that has a total distributed ground capacitance C, the total ground capacitance of the element is subdivided into three lumped capacitances to ground that are connected to the three nodes of the equivalent resistance network of the triangle. To find an appropriate distribution of the total ground capacitance over the nodes of the network, we consider the following experiment: Put one of the nodes of the triangle at a potential of 1V and the other nodes at 0V (see Figure 2.14). When C is the total distributed ground capacitance of the triangle, then the charge that is stored on the surface of the triangle is given by $\int_{\Delta^e} \phi(x,y) \frac{C}{\Delta^e} dS = C/3$. This result will be valid for all three nodes of the element. Thus, the capacitive "weight" seen on each of the nodes of the triangle with the other nodes connected to ground has a value $C/3$. This corresponds to an RC

network as given in Figure 2.15a. Therefore, we will use the RC network given in Figure 2.15a to approximate the distributed resistive and capacitive effects of a triangle.

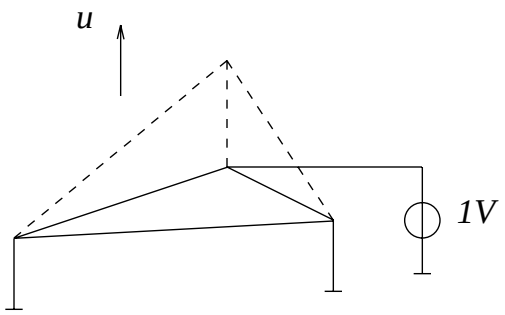


Figure 2.14. Experiment to find the lumped node capacitances of a triangle.

For a rectangle that has a total distributed ground capacitance *C*, the application of the above technique will yield an RC model where two nodes have a capacitance to ground *C/3* and the other two nodes have a capacitance to ground *C/6*. However, based on symmetry (the rectangle can be split into two triangles in two different ways), the values of these capacitances may be swapped between the two different node pairs. Therefore, for the rectangle we will use a symmetric RC model as shown in Figure 2.15b, where the total ground capacitance *C* of the rectangle is divided into four parts *C/4* that are connected to the four nodes of the equivalent resistance network of the rectangle.

For an edge with a total uniformly distributed ground capacitance *C*, a subdivision of the capacitance may be chosen as shown in Figure 2.15c. Using the principle of symmetry, both nodes that are connected to the edge receive a lumped capacitance *C/2*.

As an example of an initial RC mesh with ground capacitances, we consider the conductor shown in Figure 2.2 (see also Figure 2.7). When the conductor has a total uniformly distributed ground capacitance that is *112fF* and when $\sigma = 1\ S/\square$, the equivalent RC mesh for the interconnection according to the above discretization is given by Figure 2.16.

2.3.2.2 Coupling capacitances

For the discretization of the distributed coupling capacitance between two triangles, the discretization of the distributed coupling capacitance between two rectangles, and the discretization of the distributed coupling capacitance between two edges, we introduce without argumentation the lumped RC models shown in Figure 2.17.

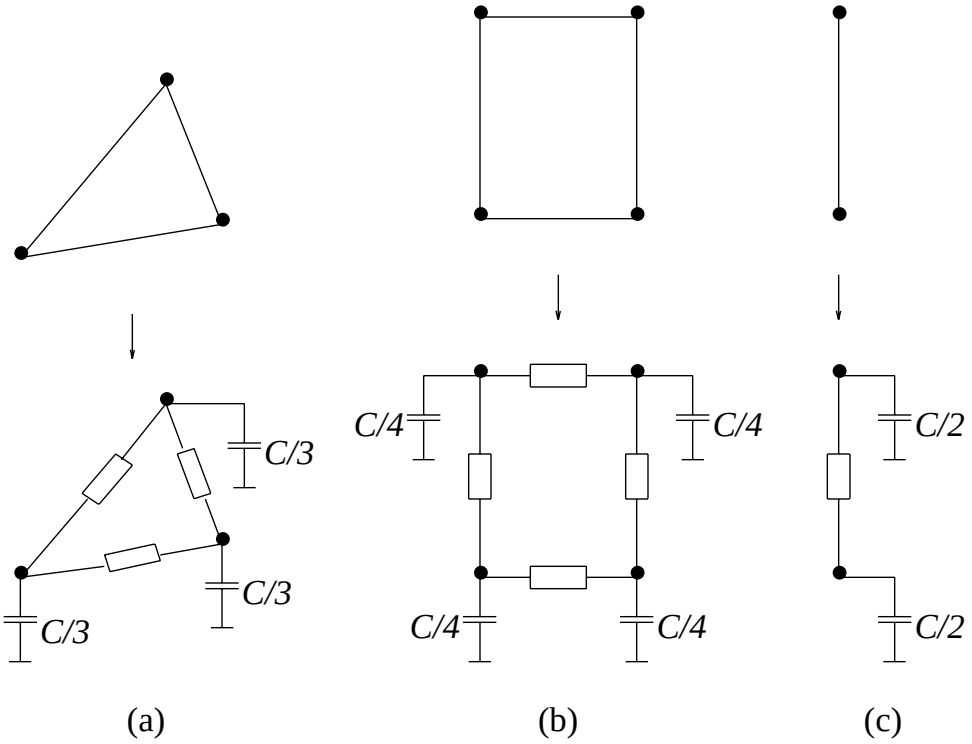


Figure 2.15. Assignment of lumped ground capacitances to the nodes of a triangle (a), a rectangle (b), and an edge (c).

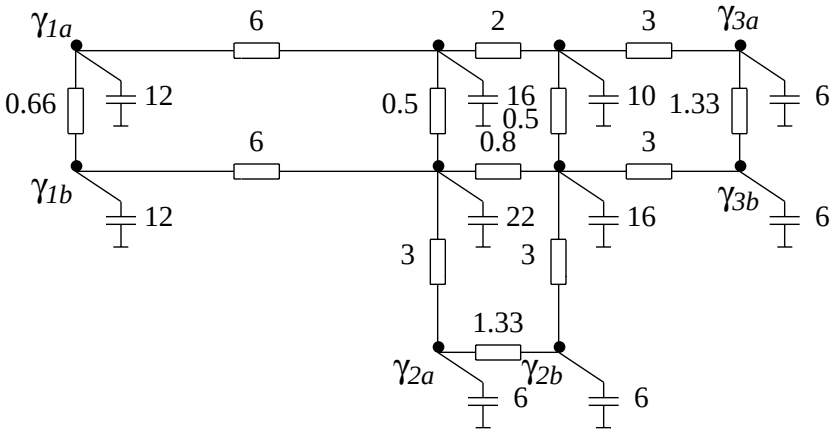


Figure 2.16. RC mesh for the conductor of Figure 2.2.

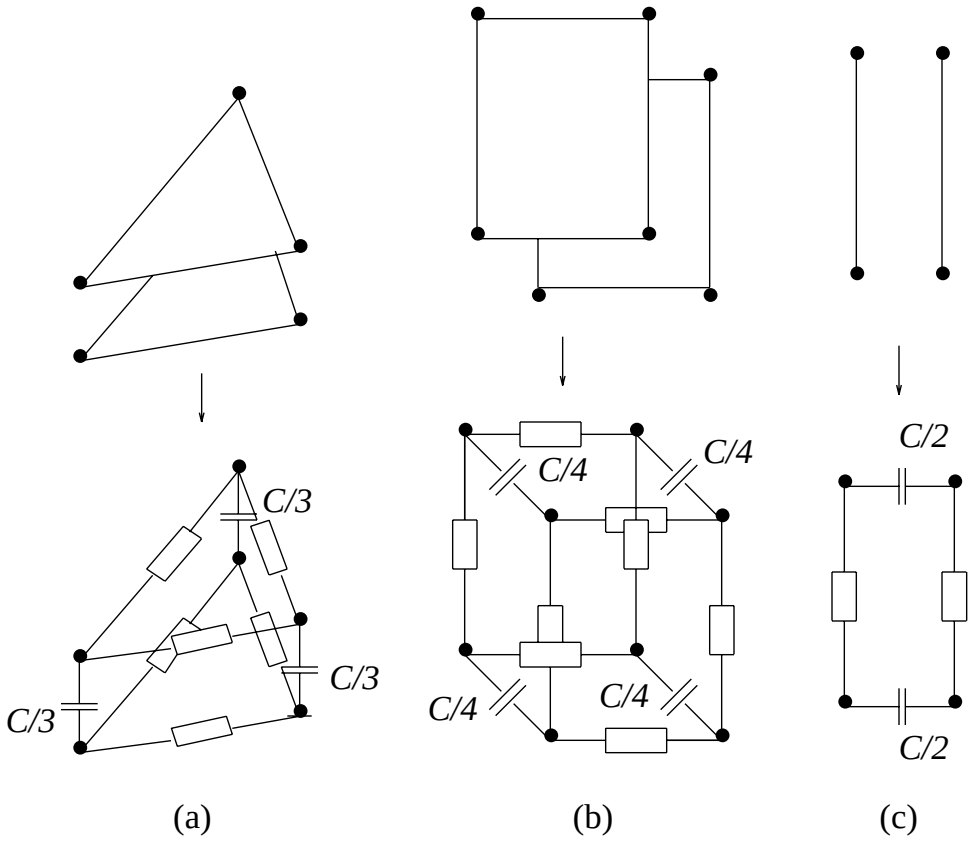


Figure 2.17. Assignment of lumped coupling capacitances to the nodes of a triangle pair (a), a rectangle pair (b), and an edge pair of a rectangle or a triangle (c).

2.3.3 The Elmore time constant

The delay times with which voltage changes are propagated through a linear network may be characterized by the value of Elmore's delay, or the Elmore time constant, between the input node and the output node of the network [27]. Preservation of the values of the Elmore time constants between the terminals of an RC network when transforming a complex RC network into a simple RC network will, therefore, assure a certain resemblance between the transmission behavior of the complex network and the transmission behavior of the simple network. This justifies the use of the technique for the simplification of the RC meshes for modeling the distributed RC effects. In this section, we will give the definition of Elmore's delay and the definition of the Elmore time constant. We will derive their properties, and we will extend the definition of the Elmore time constant as applied to RC networks representing more than one conductor and containing coupling capacitances. Based on these definitions and properties, we will elucidate the usefulness of the preservation of the Elmore time constants for

the simplification of the RC networks that are used to model the interconnections.

2.3.3.1 Definitions and properties

The original definition of Elmore's delay was given in [27]. It was used to obtain a definition of the delay in a linear network that is independent of the definition of any voltage level and that can directly be computed from the transfer function of the network. The definition of Elmore's delay given in [27] is only valid for zero initial conditions of the network and it is therefore called T_D^0. It is defined as follows:

Definition 2.1 :

Consider a linear network with zero initial conditions (i.e. the potential of each node in the network is equal to 0). When a unit voltage step is applied to an input node at $t = 0$ and when $u(t)$ is the normalized potential at an output node of the network (i.e. $u(t)$ is normalized such that $\lim_{t\to\infty} u(t) = 1$), then Elmore's delay T_D^0 between the input node and the output node is defined as

$$T_D^0 = \int_0^\infty t \, \frac{du}{dt} \, dt. \tag{2.56}$$

The relation between T_D^0 and the transfer function of the network is given by the following theorem:

Theorem 2.2 :

Let $g(s)$ be the Laplace transform of the transfer function of a linear network for the relation between the normalized potential of an input node and the normalized potential of an output node (i.e. $g(s) = v_o(s) / v_i(s)$, where $v_o(s)$ is the Laplace transform of the normalized potential at the output node and $v_i(s)$ is the Laplace transform of the normalized potential at the input node, with zero initial conditions of the network), and let that this transfer function be written as (see e.g. [29])

$$g(s) = \frac{1 + a_1 s + a_2 s^2 + \cdots + a_n s^n}{1 + b_1 s + b_2 s^2 + \cdots + b_m s^m}, \tag{2.57}$$

then the value of Elmore's delay as defined in Definition 2.1, between the input and the output of the network, is given by

$$T_D^0 = b_1 - a_1. \tag{2.58}$$

Proof:

When applying a unit step voltage to the input node at $t = 0$, with zero initial conditions of the network, the Laplace transform of $\frac{du}{dt}$ at the output node may be written as

$$\int_0^\infty \frac{du}{dt}\, e^{-st}\, dt = u(t)\, e^{-st} \Big|_0^\infty + s \int_0^\infty u(t)\, e^{-st}\, dt$$
$$= s\, \frac{g(s)}{s}, \tag{2.59}$$

which shows that $g(s)$ is found from

$$g(s) = \int_0^\infty \frac{du}{dt}\, e^{-st}\, dt. \tag{2.60}$$

Expansion of the right-hand side of 2.60 in a Taylor series yields,

$$g(s) = 1 - s \int_0^\infty t\, \frac{du}{dt}\, dt + \frac{s^2}{2!} \int_0^\infty t^2\, \frac{du}{dt}\, dt - \cdots . \tag{2.61}$$

From 2.61 we conclude that T_D^0 may be found from

$$T_D^0 = \lim_{s \to 0} \left(\frac{1}{s} - \frac{g(s)}{s} \right), \tag{2.62}$$

if this limit exists. Then, the relation between T_D^0 and $g(s)$ is established by inserting 2.57 into 2.62, which yields

$$T_D^0 = \lim_{s \to 0} \left(\frac{1}{s} - \frac{1 + a_1 s + a_2 s^2 + \cdots + a_n s^n}{s(1 + b_1 s + b_2 s^2 + \cdots + b_m s^m)} \right)$$

$$= b_1 - a_1. \tag{2.63}$$

□

The definition of T_D^0 loses its meaning for non-zero initial conditions of the network since then $\frac{du}{dt}$ may become smaller than 0 during some time interval, and the value of T_D^0 is no longer directly related to the moment when $u(t)$ stabilizes. To circumvent this problem, another definition of Elmore's delay was introduced in [30], where it is made equal to T_D^0 for zero initial conditions of the network. In

contrary with T_D^0, the definition of Elmore's delay as given in [30], also provides a useful measurement for the delay time of the network for non-zero initial conditions of the network. To indicate the difference between this latter definition and the definition of Elmore's delay given in Definition 2.1, we will call this definition of Elmore's delay T_D. The definition of T_D is as follows:

Definition 2.2 :

Consider a linear network with possibly non-zero initial conditions. When a unit voltage step is applied to an input node at $t = 0$, and when $u(t)$ is the normalized potential at an output node of the network, then Elmore's delay T_D between the input node and the output node is defined as

$$T_D = \int_0^\infty [1 - u(t)]\, dt. \tag{2.64}$$

The relation between T_D and the transfer function of the network is shown by the following theorem:

Theorem 2.3 :

Let $g(s)$ be the Laplace transform of the transfer function of a linear network for the relation between the normalized potential of an input node and the normalized potential of an output node, and let this transfer function be written as

$$g(s) = \frac{1 + a_1 s + a_2 s^2 + \cdots + a_n s^n}{1 + b_1 s + b_2 s^2 + \cdots + b_m s^m}, \tag{2.65}$$

then the value of Elmore's delay as defined in Definition 2.2 between the input and the output of the network, for zero initial conditions of the network, is given by

$$T_D = b_1 - a_1. \tag{2.66}$$

Proof:

When applying a unit step voltage to the input node at $t = 0$, with zero initial conditions of the network, the Laplace transform of $1 - u(t)$ at the output node may be written as

$$\int_0^\infty [1 - u(t)]\, e^{-st}\, dt = \frac{1}{s} - \frac{g(s)}{s}. \tag{2.67}$$

By writing the left-hand side as a Taylor series we find

$$\frac{1}{s} - \frac{g(s)}{s} = \int_0^\infty [1 - u(t)]\, dt - s \int_0^\infty t\, [1 - u(t)]\, dt + \frac{s^2}{2!} \int_0^\infty t^2\, [1 - u(t)]\, dt - \cdots . \tag{2.68}$$

From relation 2.68 it follows that T_D as defined in Definition 2.2 may be found from

$$T_D = \lim_{s \to 0} \left[\frac{1}{s} - \frac{g(s)}{s} \right], \tag{2.69}$$

if this limit exists. Then, the relation between T_D and $g(s)$ is established by inserting 2.65 into 2.69, which yields

$$T_D = \lim_{s \to 0} \left[\frac{1}{s} - \frac{1 + a_1 s + a_2 s^2 + \cdots + a_n s^n}{s(1 + b_1 s + b_2 s^2 + \cdots + b_m s^m)} \right]$$

$$= b_1 - a_1. \tag{2.70}$$

□

Thus, for zero initial conditions of the network it holds that $T_D^0 = T_D = b_1 - a_1$. For an RC network the value of Elmore's delay T_D^0 (or T_D for zero initial conditions) can be calculated directly from a network constant which is denoted by τ and which is called the Elmore time constant (also: first-order time constant) of the network. This has been described in [31] for the case where the RC network is a tree network and where it contains no coupling capacitances. In [30] and [32] this result was extended for RC networks that also contain resistance loops. Below, we will describe how the Elmore time constant is computed for RC networks containing resistance loops as well as coupling capacitances. First, we will treat the case where the RC network represents a single conductor. Next, we will present a definition for the Elmore time constant for an RC network that represents more than one conductor, and where coupling capacitances are also present between two nodes that are on a different conductor. We will show that this definition of τ is equal to the value of Elmore's delay also for these types of networks.

In order to obtain the desired definitions and properties we first introduce the following definition (see also Figure 2.18):

Definition 2.3 :

Consider a linear resistance network with nodes $1 \cdots N$, in which there is at least one path via resistances between each pair of nodes. Let $u_1 \cdots u_N$ represent the voltages of the nodes in the network and let $i_1 \cdots i_N$ represent the currents that are flowing into the nodes of the resistance network. The potential of node j is held fixed at a potential u_j. Then, R_{kij} for each node triplet k, i and j $\{ k, i, j \mid k = 1 \cdots N, i = 1 \cdots N, j = 1 \cdots N \}$ is defined as

$$R_{kij} = \frac{u_i - u_j}{i_k} \qquad i_l = 0, \;\; i_k \neq 0, \;\; (l = 1 \cdots N, l \neq k). \tag{2.71}$$

Using Definition 2.3 it is possible to define the Elmore constant τ_{ij} between a node i and a node j in an RC network representing a single conductor as:

Definition 2.4 :

Consider a linear RC network with nodes $1 \cdots N$, in which each node k $\{ k \mid k = 1 \cdots N \}$ has a linear ground capacitance C_k connected it, in which each node pair k, l $\{ k, l \mid k = 1 \cdots N , l = 1 \cdots N \}$ has a linear coupling capacitance C_{kl}, and in which there is at least one path via resistances between two nodes in the network, Then, the Elmore time constant between a node i and a node j in the network is defined as

$$\tau_{ij} = \sum_{k=1}^{N} R_{kij} \, C_k, \tag{2.72}$$

where R_{kij} is given by Definition 2.3,

The relation between τ_{ij} and T_D^0 (or T_D for zero initial conditions of the network) then follows from the following theorem:

Theorem 2.4 :

Consider a linear RC network with nodes $1 \cdots N$, in which each node k $\{ k \mid k = 1 \cdots N \}$ has a linear ground capacitance C_k connected it, and in which each node pair k, l $\{ k, l \mid k = 1 \cdots N , l = 1 \cdots N \}$ has a linear coupling capacitance C_{kl}. Then, Elmore's delay T_{Dij}^0 (or T_{Dij} for zero initial conditions of the network) for the transmission of signals from node j to node i is found from

$$T_{Dij}^0 = \tau_{ij}, \tag{2.73}$$

where τ_{kij} is given by Definition 2.4.

Proof:

From the definition of R_{kij} it follows via superposition of the contributions of each of the currents $i_1 \cdots i_N$ that

$$u_i(t) - u_j(t) = \sum_{k=1}^{N} R_{kij}\, i_k(t). \tag{2.74}$$

When a unit step voltage is applied to node j this leads to the relation

$$1 - u_i(t) = \sum_{k=1}^{N} - R_{kij}\, i_k(t). \tag{2.75}$$

Using $T_D^0 = T_D$ (this is appropriate since we use zero initial conditions), substitution of 2.75 in 2.64 gives

$$\begin{aligned} T_{Dij}^0 &= \int_0^\infty \sum_{k=1}^{N} - R_{kij}\, i_k(t)\, dt \\ &= \int_0^\infty \sum_{k=1}^{N} R_{kij} \left[C_k \frac{du_k}{dt} + \sum_{l=1}^{N} C_{kl} \left(\frac{du_k}{dt} - \frac{du_l}{dt} \right) \right] dt \\ &= \sum_{k=1}^{N} R_{kij}\, C_k \\ &= \tau_{ij}. \end{aligned} \tag{2.76}$$

□

As an example of the calculation of the Elmore time constant for a pair of nodes in an RC tree network, we consider the network shown in Figure 2.19. For a tree network there is only one path via resistances between two nodes. Hence it follows from Definition 2.3 that R_{kij} for the node triplet i, j, k in a tree network is given the sum of the resistances on the part of the path to node j that is shared between node i and node k. Based on this notion, it is then easily found that the value of the Elmore time constant between node 1 and node 4 in Figure 2.19 has the value shown in Figure 2.19.

To obtain a definition of the Elmore time constant that is also valid for an RC network consisting of different conductors, we give the following definition:

Definition 2.5 :

Consider a linear RC network with nodes $1 \cdots N$, in which each node k $\{ k \mid k = 1 \cdots N \}$ has a linear ground capacitance C_k connected it, and in

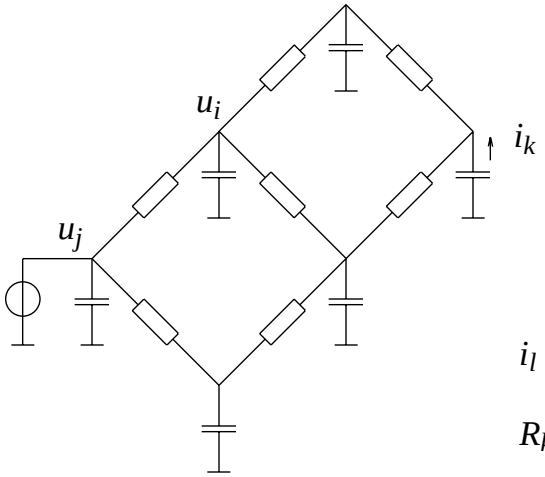
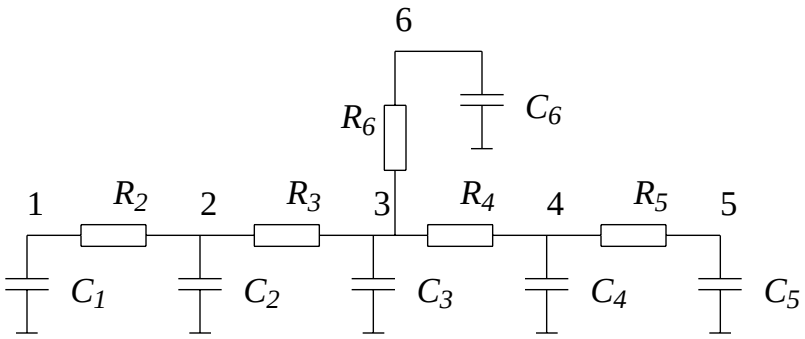


Figure 2.18. Definition of R_{kij}.



$$T_{D4,1} = R_2\ C_2 + (\ R_2 + R_3\ )\ C_3 + (\ R_2 + R_3 + R_4\ )\ C_4 + (\ R_2 + R_3 + R_4\ )\ C_5 + (\ R_2 + R_3\ )\ C_6$$

Figure 2.19. Example of the determination of T_{Dij} for an RC tree.

which each node pair k, l $\{\ k, l \mid k = 1\ \cdots\ N\ ,\ l = 1\ \cdots\ N\ \}$ has a linear coupling capacitance C_{kl}. The nodes are subdivided into clusters of nodes $\{1\ \cdots\ n_1\}$, $\{n_1 + 1\ \cdots\ n_2\}$, ... $\{n_{C-1} + 1\ \cdots\ N\}$ such that two nodes of the same cluster are connected to each other via one or more resistances, and such that there is no connection along resistances between two nodes of a different cluster. (Each cluster is also called a conductor.) Then, we define the Elmore time constant between two nodes i and j that are part of the cluster $\{n_{c-1}+1\ \cdots\ n_c\}$ as

$$\tau_{ij} = \sum_{k=n_{c-1}+1}^{n_c} R_{kij} \left[C_k + \sum_{l=1,\ l \neq n_{c-1}+1 \ \cdots \ n_c}^{N} C_{kl} \right], \tag{2.77}$$

where R_{kij} is defined according to Definition 2.3 for the sub-network consisting of nodes $\{n_{c-1}+1 \ \cdots \ n_c\}$.

The relation with T^0_{Dij} follows from the following theorem:

Theorem 2.5 :

Consider a linear RC network with nodes $1 \ \cdots \ N$, in which each node k $\{ k \mid k = 1 \ \cdots \ N \}$ has a ground capacitance C_k connected it, and in which each node pair k, l $\{ k, l \mid k = 1 \ \cdots \ N, l = 1 \ \cdots N \}$ has a coupling capacitance C_{kl} between them. The nodes are subdivided into clusters of nodes $\{1 \ \cdots \ n_1\}$, $\{n_1 + 1 \ \cdots \ n_2\}$, ... $\{n_{C-1} + 1 \ \cdots \ N\}$ such that two nodes of the same cluster are connected to each other via one or more resistances, and such that there is no connection along resistances between two nodes of a different cluster. Then, when $n_{c-1} + 1 \ \cdots \ n_c$ is the cluster of nodes of which i and j are part of, and for the other clusters it holds that at least one node of each cluster is connected to a voltage source that has a fixed potential, Elmore's delay T^0_{Dij} (or T_{Dij} for zero initial conditions of the network) for the transmission of signals from node j to node i is found from

$$T^0_{Dij} = \tau_{ij}, \tag{2.78}$$

where τ_{ij} is defined according to Definition 2.5

Proof:

From the definition of R_{kij} it follows via superposition of the contributions of each of the currents $i_1 \ \cdots \ i_N$ that

$$u_i(t) - u_j(t) = \sum_{k=n_{c-1}+1}^{n_c} R_{kij} \ i_k(t). \tag{2.79}$$

When a unit step voltage is applied to node j this leads to the relation

$$1 - u_i(t) = \sum_{k=n_{c-1}+1}^{n_c} - R_{kij} \ i_k(t). \tag{2.80}$$

Since $T^0_D = T_D$ for zero initial conditions, substitution of 2.80 in 2.64 gives

$$
\begin{aligned}
T^0_{Dij} &= \int_0^\infty \sum_{k=n_{c-1}+1}^{n_c} - R_{kij}\, i_k(t)\, dt \\
&= \int_0^\infty \sum_{k=n_{c-1}+1}^{n_c} R_{kij} \left[C_k \frac{du_k}{dt} + \sum_{l=1}^{N} C_{kl} \left(\frac{du_k}{dt} - \frac{du_l}{dt} \right) \right] dt \\
&= \sum_{k=n_{c-1}+1}^{n_c} R_{kij} \left[C_k + \sum_{l=1,\, l \neq n_{c-1}+1 \,\cdots\, n_c}^{N} C_{kl} \right] \\
&= \tau_{ij}. \qquad (2.81)
\end{aligned}
$$

Remark that $u_l(\infty) = u_l(0)$ for $\{ l \mid l = 1 \cdots N, l \neq n_{c-1}+1 \cdots n_c \}$ since the conductors other than the conductor of which *i* and *j* are part of are connected to a voltage source that has a fixed potential.

□

2.3.3.2 Use for network reduction

According to the previous discussion an Elmore time constant can be associated with each directed pair of terminals in a linear RC network. When transforming the complex RC network that is used to model the distributed RC effects of the interconnections into a simple RC network, the equality between the values of the Elmore time constants between the terminals of the complex network, and the values of the Elmore time constants between the corresponding terminals of the simple network, may be used as a condition to derive the simple network from the complex network. This is achieved by requiring for the Elmore time constant τ_{ij} between two terminals *i, j* in the complex network and for the Elmore time constant τ'_{ij} for the corresponding pair of terminals in the simple network

$$\tau_{ij} = \tau'_{ij}. \qquad (2.82)$$

The resemblance between the transmission behavior of the complex network and the simple network then follows from Definition 2.1, Definition 2.2, Theorem 2.4 and Theorem 2.5:

From 2.82, Definition 2.1 and Theorem 2.4 or Theorem 2.5 it follows that, for zero initial conditions, when applying a unit step voltage to some terminal *j* in the complex network at $t = 0$, and when applying the same signal to the corresponding terminal in the simple network, for the potential $u_i(t)$ of a terminal *i* in the complex network and for the potential $u_i'(t)$ of the corresponding terminal in the simple network, it will hold that

$$\int_0^\infty t \frac{du}{dt}\, dt = \int_0^\infty t \frac{du'}{dt}\, dt \tag{2.83}$$

or

$$\int_0^\infty t \left[\frac{du}{dt} - \frac{du'}{dt}\right] dt = 0. \tag{2.84}$$

From 2.82, Definition 2.2 and Theorem 2.4 or Theorem 2.5 it follows that, under the same circumstances as described above, for the potential $u_i(t)$ of a terminal i in the complex network and for the potential $u_i'(t)$ of the corresponding terminal in the simple network, it will hold that

$$\int_0^\infty [1 - u_i(t)]\, dt = \int_0^\infty [1 - u_i'(t)]\, dt \tag{2.85}$$

or

$$\int_0^\infty [u_i(t) - u_i'(t)]\, dt = 0. \tag{2.86}$$

As an example of a network reduction while preserving the Elmore time constants, and its influence on the transmission behavior of the network, we consider the derivation of a lumped RC for a one-dimensional uniformly distributed RC line as shown in Figure 2.20. The RC line has a length L, a total resistance R and total distributed ground capacitance C. The Elmore time constants follow from

$$\begin{aligned} \tau_{ij} = \tau_{ji} &= \int_0^L r(x)\, dc(x) \\ &= \int_0^L \frac{R}{L} x \cdot \frac{C}{L}\, dx \\ &= \frac{1}{2} RC. \end{aligned} \tag{2.87}$$

Hence an appropriate RC model in which the values of the Elmore time constants between the input nodes are preserved, in which the total capacitance to ground is preserved, and in which the resistance between the two terminals in the model is equal to the total resistance between the two terminals of the distributed line, is given by the π-model shown in Figure 2.20b. A comparison of the step response behavior of the lumped π-model and the step response behavior of the distributed RC line is shown in Figure 2.21.

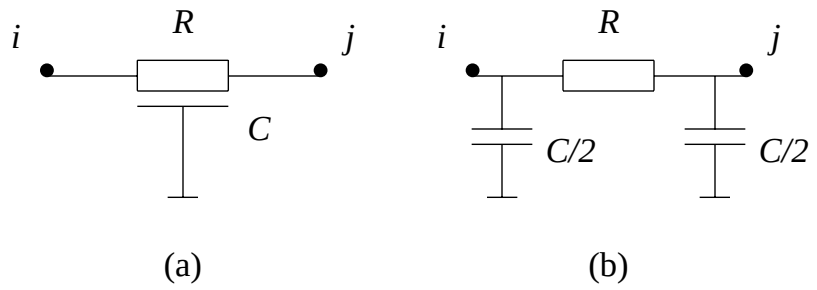


Figure 2.20. A distributed RC line (a) and its π-model (b).

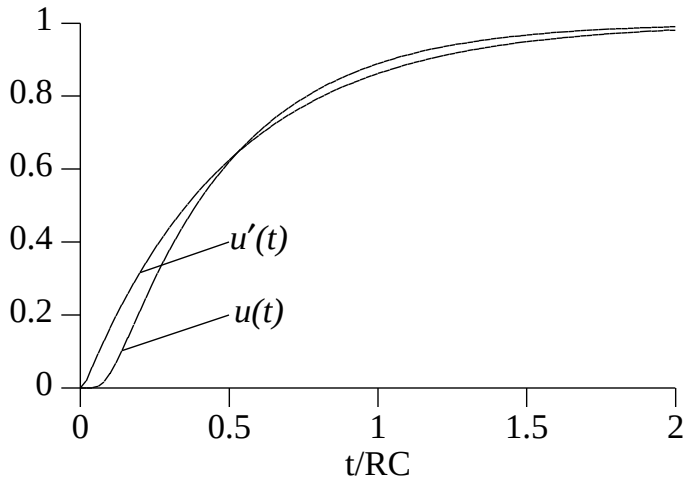


Figure 2.21. Voltage step response behavior of the distributed RC line in Figure 2.20a (*u(t)*) and the π-model in Figure 2.20b (*u′(t)*).

2.3.4 RC network reduction

To guarantee a close resemblance between the electrical properties of the initial RC network and the electrical properties of the final RC network, we will satisfy the following conditions when transforming the initial RC network into the final RC network:

a. The relation between the steady-state potentials and the steady-state currents of the terminals of the final network is the same as for the terminals of the initial network.

b. The total capacitance to ground for each of the conductors is unchanged and the total coupling capacitance between two conductors is unchanged.

c. For each directed pair of nodes that remain in the final network and that are on the same conductor, the value of the Elmore time constant is preserved.

ad a)
This condition will guarantee that the steady-state behavior of the initial network is equal to the steady-state behavior of the final network.

ad b)
This condition will assure that the charge storing capacity of the initial network is equal to the charge storing capacity of the final network.

ad c)
See Section 2.3.3.

As we will show in the following, it appears that it is possible to satisfy all three conditions for an arbitrarily RC network as discussed in Section 2.3.2. The desired network reduction is achieved by an algorithm that repeatedly eliminates a node from the network until a network is obtained that has the same number of nodes as the number of nodes that is required in the final network. By satisfying the conditions (a), (b) and (c) during each elimination step, the conditions (a), (b) and (c) are then guaranteed for the total network reduction. The algorithm that performs this network reduction has been described in a simplified form in [16]. One step of the algorithm is defined as follows:

Definition 2.6 :
Consider a linear RC network with nodes $1 \cdots N$, in which each node i $\{ i \mid i = 1 \cdots N \}$ has a linear ground capacitance C_i connected it, and in which each node pair i, j $\{ i, j \mid i = 1 \cdots N , j = 1 \cdots N, i \neq j \}$ has a linear coupling capacitance C_{ij}, and a conductance G_{ij}. Let the nodes be subdivided into clusters of nodes $\{1 \cdots n_1\}$, $\{n_1 + 1 \cdots n_2\}$, ... $\{n_{C-1} + 1 \cdots N\}$ such that two nodes of the same cluster are connected to each other via one or more resistances (i.e. non-zero conductances), and such that there is no connection along resistances between two nodes of a different cluster. (Each cluster is also called a conductor.) Let the admittance matrix for the relation between the node currents $I(s)$ and the node potentials $\Phi(s)$ be denoted by $Y(s)$ (i.e. $I(s) = Y(s)\,\Phi(s)$), where each element Y_{ij} of $Y(s)$ is given by

$$Y_{ij} = A_{ij} + sB_{ij}, \tag{2.88}$$

with

$$A_{ij} = -G_{ij} \qquad (i \neq j) \tag{2.89a}$$
$$B_{ij} = -C_{ij} \qquad (i \neq j) \tag{2.89b}$$

$$A_{ii} = \sum_{j=1,\, j\neq i}^{N} G_{ij} \tag{2.89c}$$

$$B_{ii} = C_i + \sum_{j=1,\, j\neq i}^{N} C_{ij}. \tag{2.89d}$$

Then, the elimination of a node k from the RC network is defined such that the entries $Y_{ij}^{(2)} = A_{ij}^{(2)} + sB_{ij}^{(2)}$ $\{ i, j \mid i = 1 \cdots N, j = 1 \cdots N \}$ of the admittance matrix $Y^{(2)}(s)$ of the new RC network are found from

$$A_{ik}^{(2)} + sB_{ik}^{(2)} = 0 \tag{2.90a}$$

$$A_{kj}^{(2)} + sB_{kj}^{(2)} = 0 \tag{2.90b}$$

$$A_{ij}^{(2)} + sB_{ij}^{(2)} = A_{ij} + sB_{ij} - \left(\frac{A_{kj}A_{ik} + sA_{kj}B_{ik} + sA_{ik}B_{kj}}{A_{kk}} \right) \quad (i \neq j \wedge i \neq k \wedge j \neq k) \tag{2.90c}$$

$$A_{ii}^{(2)} + sB_{ii}^{(2)} = A_{ii} + sB_{ii} - \left(\frac{A_{ki}^2 + 2sA_{ki}B_{ki} - sA_{ik}B_{kk}}{A_{kk}} \right) \quad (i \neq k). \tag{2.90d}$$

The above corresponds to a new RC network with nodes $\{ i \mid i = 1 \cdots N, i \neq k \}$, conductances $G_{ij}^{(2)}$ $\{ i, j \mid i = 1 \cdots N, j = 1 \cdots N, i \neq k \wedge j \neq k \}$, ground capacitances $C_i^{(2)}$ $\{ i \mid i = 1 \cdots N, i \neq k \}$ and coupling capacitances $C_{ij}^{(2)}$ $\{ i, j \mid i = 1 \cdots N, j = 1 \cdots N, i \neq k \wedge j \neq k \}$, that are given by

$$G_{ij}^{(2)} := G_{ij} + \frac{G_{kj}\, G_{ik}}{\sum_{l=1,\, l\neq k}^{N} G_{kl}} \tag{2.91a}$$

$$C_i^{(2)} := C_i + \frac{G_{ki}}{\sum_{l=1,\, l\neq k}^{N} G_{kl}} C_k. \tag{2.91b}$$

$$C_{ij}^{(2)} := C_{ij} + \frac{G_{kj}}{\sum_{l=1,\, l\neq k}^{N} G_{kl}} C_{ki} + \frac{G_{ki}}{\sum_{l=1,\, l\neq k}^{N} G_{kl}} C_{kj} \qquad (i \neq j). \tag{2.91c}$$

The fulfillment of the conditions (a), (b) and (c) is shown by the following three theorems:

Theorem 2.6 :

For the elimination step as given by Definition 2.6, the relation between the steady-state potentials and the steady-state currents of the terminals of the original network is the same as for the terminals of the reduced network.

This theorem is easily verified from Definition 2.6 by noting that the new conductances in the reduced network are found from the original network by means of Gaussian elimination (see 2.91a).

Theorem 2.7 :

For the elimination step as given by Definition 2.6, the total capacitance to ground for each of the conductors is unchanged and the total coupling capacitance between two conductors is unchanged.

Proof:

The preservation of the ground capacitances follows from 2.91b by summing the increments of the ground capacitances C_i ($i = 1 \cdots N,\ i \neq k$):

$$\sum_{i=1,\ i\neq k}^{N} C_i^{(2)} - C_i = \sum_{i=1,\ i\neq k}^{N} \frac{G_{ki}}{\sum_{l=1,\ l\neq k}^{N} G_{kl}} C_k = C_k, \tag{2.92}$$

so that

$$\sum_{i=1,\ i\neq k}^{N} C_i^{(2)} = \sum_{i=1,}^{N} C_i. \tag{2.93}$$

The preservation of each of the coupling capacitances between the conductors follows from an investigation of the set of coupling capacitances $\{ C_{ij} \mid i = n_{I-1}+1 \cdots n_{I,}\ j = n_{J-1}+1 \cdots n_J \}$ that are present between two arbitrary conductors I and J. If k is not on conductor I nor on conductor J, it follows from 2.91c (since $G_{kj} = 0$ and $G_{ki} = 0$) that

$$C_{ij}^{(2)} - C_{ij} = 0 \quad (i = n_{I-1}+1 \cdots n_I,\ j = n_{J-1}+1 \cdots n_J). \tag{2.94}$$

If k is either on conductor I or on conductor J - say k is on conductor J in this case - we have from 2.91c

$$\sum_{j=n_{J-1}+1,\ j\neq k}^{n_J} C_{ij}^{(2)} - C_{ij} = \sum_{J=n_{J-1}+1,\ j\neq k}^{n_J} \frac{G_{kj}}{\sum_{l=1,\ l\neq k}^{N} G_{kl}} C_{ki} = C_{ki},$$
$$(i = n_{I-1}+1 \ \cdots \ n_I), \quad (2.95)$$

so that

$$\sum_{i=n_{I-1}+1,\ }^{n_I} \sum_{j=n_{J-1}+1,\ j\neq k}^{n_J} C_{ij}^{(2)} = \sum_{i=n_{I-1}+1,\ }^{n_I} \sum_{j=n_{J-1}+1}^{n_J} C_{ij}. \quad (2.96)$$

Thus, in both cases the total coupling capacitance between conductor I and conductor J is preserved.

□

In order to show that the Elmore time constants are preserved for node pairs that are on the same conductor and that remain in the reduced network, we first introduce the following property:

Lemma 2.1 :

Consider a linear network that has nodes $1 \ \cdots \ N$, a conductance G_{ij} between each node pair i, j $\{ i, j \mid i = 1 \ \cdots \ N, j = 1 \ \cdots \ N, i \neq j \}$, and in which there is at least one path via resistances between each pair of nodes. Then, it will hold for each node triplet i, j, k $\{ i, j, k \mid i = 1 \ \cdots \ N, \ j = 1 \ \cdots \ N, \ k = 1 \ \cdots \ N, \ i \neq j \wedge i \neq k \wedge j \neq k \}$,

$$\sum_{l=1,\ l\neq k}^{N} R_{lij} \frac{G_{kl}}{\sum_{m=1,\ m\neq k}^{N} G_{km}} = R_{kij}, \quad (2.97)$$

where the value of R_{kij} is given by Definition 2.3.

Proof:

Assume an equilibrium state of the network (potentials and currents are constant) and suppose

$$u_k \neq 0,$$
$$u_l = 0 \qquad (l = 1 \ \cdots \ N, l \neq k). \quad (2.98)$$

Then it follows from the current balances

$$u_l \sum_{i=1,\ i\neq l}^{N} G_{il} - \sum_{i=1,\ i\neq l}^{N} G_{il}\, u_i = i_l \qquad (l = 1 \ \cdots \ N) \quad (2.99)$$

that the currents i_l ($l = 1 \cdots N, l \neq k$) are found from

$$i_l = \frac{G_{kl}}{\sum_{m=1,\, m \neq k}^{N} G_{km}} \, i_k \qquad (l = 1 \cdots N, l \neq k). \tag{2.100}$$

From the definition of R_{lij} we have for a node pair i, j $\{ i, j \mid i = 1 \cdots N, j = 1 \cdots N, i \neq j \wedge i \neq k \wedge j \neq k \}$,

$$u_i - u_j = 0 = \sum_{l=1}^{N} R_{lij} \, i_l. \tag{2.101}$$

Substitution of 2.100 in 2.101 for $l = 1 \cdots N, l \neq k$ yields

$$\sum_{l=1,\, l \neq k}^{N} R_{lij} \, \frac{G_{kl}}{\sum_{m=1,\, m \neq k}^{N} G_{km}} \, i_k + R_{kij} \, i_k = 0, \tag{2.102}$$

or, after rearranging terms and division by i_k,

$$\sum_{l=1,\, l \neq k}^{N} R_{lij} \, \frac{G_{kl}}{\sum_{m=1,\, m \neq k}^{N} G_{km}} = R_{kij}. \tag{2.103}$$

□

Using lemma 2.1 it is then possibly to prove the following theorem:

Theorem 2.8 :

For the elimination step as given by Definition 2.6, for each directed pair of nodes that remain in the new network and that are on the same conductor, the value of the Elmore time constant is preserved.

Proof:

To prove that the Elmore time constants are unchanged we consider a pair of nodes i, j ($i \neq j$) that are part of a cluster c with nodes $n_{c-1}+1 \cdots n_c$. The node that is eliminated, k, is either part of cluster (conductor) c, or not, and $k \neq i$ and $k \neq j$. In analogy with the definition of the Elmore constant for networks with more than one conductor, we assume all clusters but cluster c to be connected to the ground potential. Further, we introduce

$$\bar{C}_l = C_l + \sum_{m=1,\ m \neq n_{c-1}+1\ \cdots\ n_c}^{N} C_{lm} \qquad (l = n_{c-1}+1\ \cdots\ n_c), \tag{2.104}$$

so that the value of the Elmore time constant τ_{ij} between node i and node j before elimination of node k (see Definition 2.5) is given by

$$\tau_{ij} = \sum_{l=n_{c-1}+1}^{n_c} R_{lij}\, \bar{C}_l. \tag{2.105}$$

For the evaluation of the Elmore time constant between node i and node j after the elimination of node k we introduce

$$\bar{C}_l^{(2)} = C_l^{(2)} + \sum_{m=1,\ m \neq n_{c-1}+1\ \cdots\ n_c}^{N} C_{lm}^{(2)} \qquad (l = n_{c-1}+1\ \cdots\ n_c). \tag{2.106}$$

Then, we consider two different cases:

Case I: node k is in a different cluster than nodes i, j.

In this case it follows from 2.91 that the value of the Elmore constant τ'_{ij} between node i and node j after elimination of node k, is given by

$$\begin{aligned} \tau'_{ij} &= \sum_{l=n_{c-1}+1}^{n_c} R_{lij}\, \bar{C}_l^{(2)} \\ &= \sum_{l=n_{c-1}+1}^{n_c} R_{lij}\, \bar{C}_l. \end{aligned} \tag{2.107}$$

(Note that the R_{lij}s are unchanged since they are not affected by Gaussian elimination.) Thus, with 2.105 it follows that $\tau'_{ij} = \tau_{ij}$.

Case II: node k is in the same cluster as nodes i, j.

In this case it follows from 2.91 that that the value of the Elmore constant τ'_{ij} between node i and node j after elimination of node k, is given by

$$\begin{aligned} \tau'_{ij} &= \sum_{l=n_{c-1}+1,\ l \neq k}^{n_c} R_{lij}\, \bar{C}_l^{(2)} \\ &= \sum_{l=n_{c-1}+1,\ l \neq k}^{n_c} R_{lij} \left[\bar{C}_l + \frac{G_{kl}}{\sum_{m=n_{c-1}+1,\ m \neq k}^{n_c} G_{km}}\, \bar{C}_k \right]. \end{aligned} \tag{2.108}$$

Using Lemma 2.1, Equation 2.108 may be rewritten as

$$\tau'_{ij} = \sum_{l=n_{c-1}+1,\ l \neq k}^{n_c} R_{lij}\,\bar{C}_l \;+\; R_{kij}\,\bar{C}_l$$

$$= \sum_{l=n_{c-1}+1}^{n_c} R_{lij}\,\bar{C}_l. \tag{2.109}$$

Thus, in this case with 2.105 it also follows that $\tau'_{ij} = \tau_{ij}$.

Therefore, $\tau'_{ij} = \tau_{ij}$ in both cases.

□

The elimination of a node is illustrated in Figure 2.22. For the example shown in Figure 2.22 there is only one capacitor connected to the node. When there is more than one capacitor connected to the node, the other capacitors are dealt with in the same way. The elimination shown in Figure 2.22 is executed in two steps. First, the capacitance C_k that is connected to node k is distributed over the nodes i that are adjacent to node k:

$$C_i = \frac{G_{ik}}{\sum_{j=1,\ j \neq k}^{N} G_{kj}}\, C_k. \tag{2.110}$$

Next, the node k is eliminated by means of Gaussian elimination:

$$G_{ij} = \frac{G_{ik}\, G_{jk}}{\sum_{j=1,\ j \neq k} G_{kj}} \tag{2.111}$$

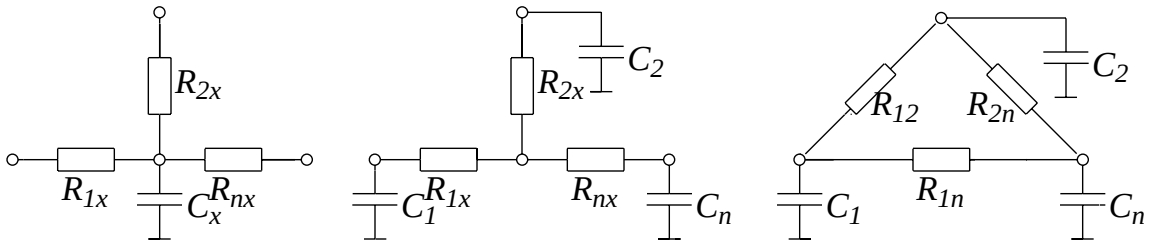


Figure 2.22. Elimination of a node k.

When applying the network reduction algorithm to the RC network of Figure 2.16, the final RC model that is shown in Figure 2.23 is obtained.

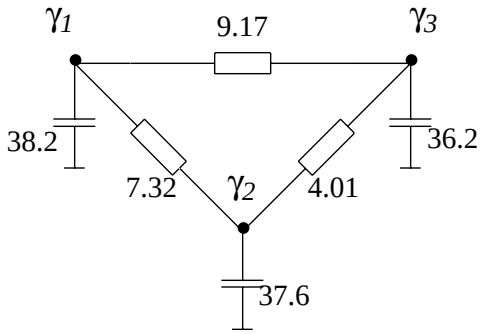


Figure 2.23. Final model for the RC network in Figure 2.16.

2.3.5 Examples

Using the preceding RC modeling method, RC models have been computed for the interconnections polygons of Figure 2.10a-c. The results are given in Table 2.4, Table 2.5 and Table 2.6. The sheet resistivity is *1* Ω/□ and the total capacitance of each interconnection polygon is *1 F*. For each of the particular interconnection polygons, we see from Table 2.4, Table 2.5 and Table 2.6 that the values of the RC model and the values of the Elmore time constants converge to a certain value as the complexity of the finite-element mesh increases.

Table 2.4. RC model and time constants for Figure 2.10a.

nodes	R_{ab} in Ω	C_a in F	C_b in F	τ_{ab}	τ_{ba}
7	2.167	0.483	0.517	1.046	1.121
8	2.286	0.500	0.500	1.143	1.143
15	2.388	0.500	0.500	1.194	1.194
37	2.470	0.500	0.500	1.235	1.235
106	2.540	0.500	0.500	1.270	1.270
304	2.550	0.500	0.500	1.275	1.275

Table 2.5. RC model and time constants for Figure 2.10b.

nodes	R_{ab} in Ω	C_a in F	C_b in F	τ_{ab}	τ_{ba}
5	1.778	0.639	0.361	1.136	0.642
6	2.133	0.630	0.369	1.344	0.787
16	2.294	0.597	0.403	1.370	0.924
36	2.350	0.588	0.412	1.382	0.968
149	2.406	0.578	0.421	1.391	1.013
375	2.455	0.570	0.429	1.399	1.053

Table 2.6. RC model and time constants for Figure 2.10c.

nodes	R_{ab} in Ω	C_a in F	C_b in F	τ_{ab}	τ_{ba}
7	2.082	0.345	0.655	0.719	1.363
8	2.082	0.344	0.656	0.716	1.366
14	2.126	0.342	0.658	0.727	1.399
41	2.199	0.336	0.664	0.740	1.460
136	2.236	0.334	0.666	0.747	1.490
342	2.245	0.334	0.666	0.749	1.496

An example that illustrates the accuracy of the network reduction technique with respect to its influence on the transmission behavior of the network is given by the piece of VLSI interconnect shown in Figure 2.24. The lumped RC model that is found for it is also shown in Figure 2.24.

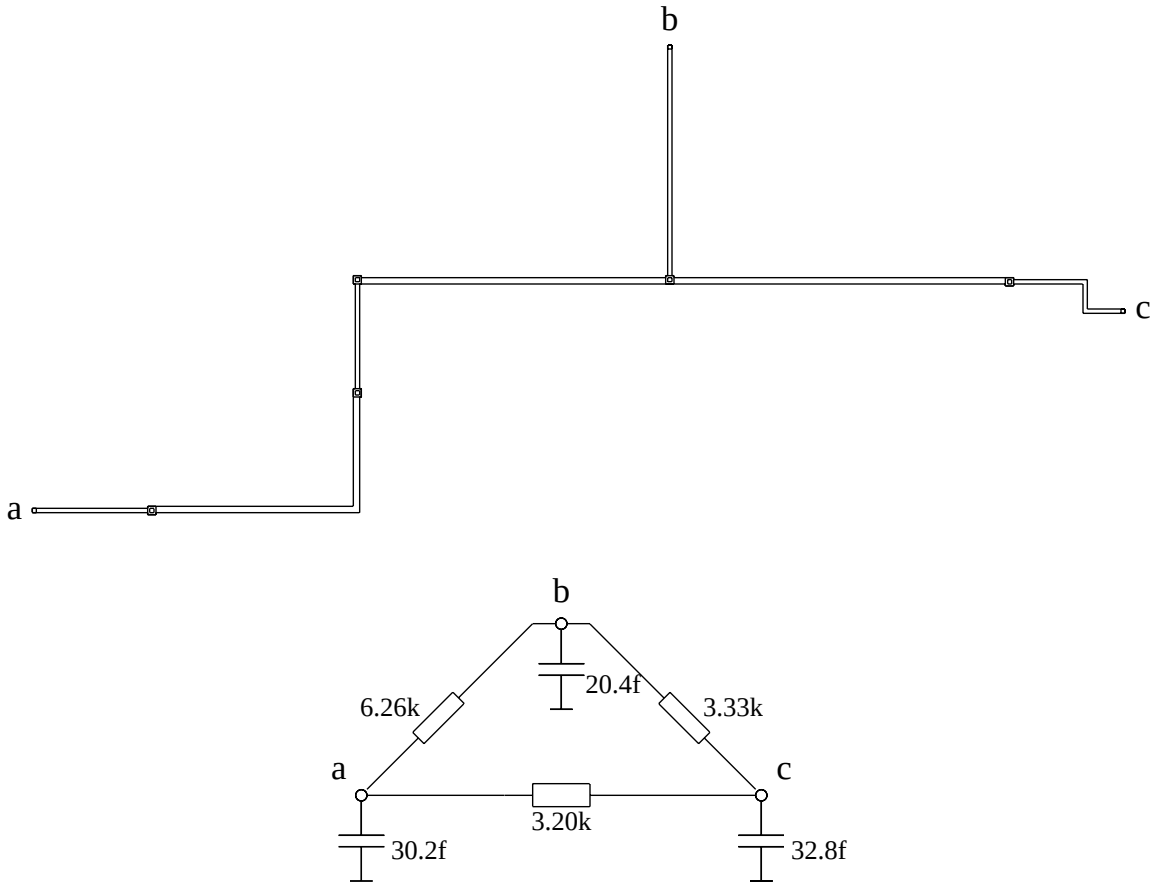


Figure 2.24. A VLSI interconnection and its extracted lumped RC model.

In Figure 2.25 the output voltage at node 'c' is plotted (dotted curve) when the RC model is simulated by using the circuit simulator SPICE, and when a ramp input voltage is applied to node 'a'. The exact solution for the voltage at node 'a' -

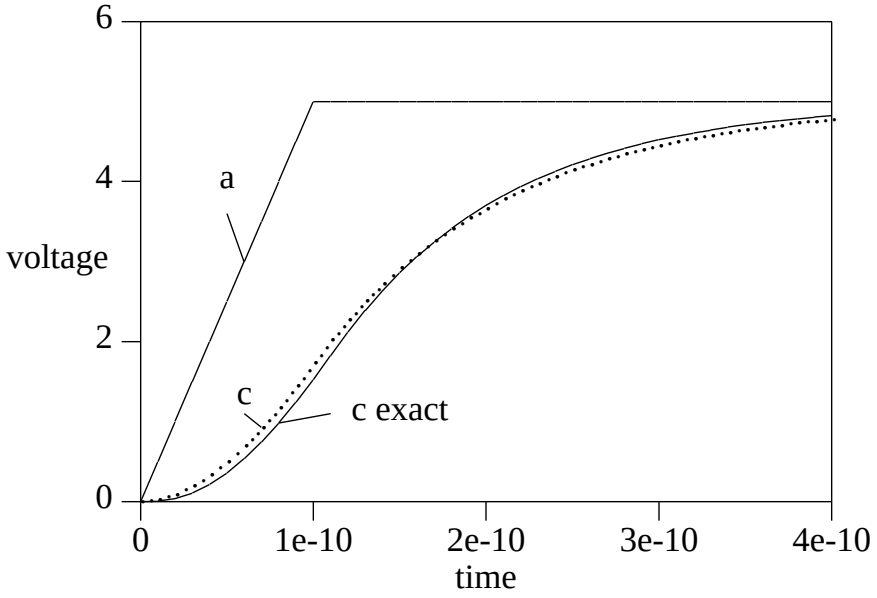


Figure 2.25. Voltage of node ’c’ when a ramp input voltage is applied to node ’a’.

which has been approximated by simulating a model for which nodes have been retained at each contact - is also given in Figure 2.25 (solid curve). A comparison between the results shows that a reduction of the number of nodes from 8 in the detailed model to 3 in the extracted model has only a small influence on the output voltage waveform. Especially around 2-3 Volt, which is important as it is the switching region for transistors, the voltage waveform of the extracted interconnection model closely matches the exact voltage waveform.

In Figure 2.26, the voltage of node ’a’ of the extracted RC model together with the exact solution is shown when a ramp input voltage is applied to node ’b’. The same conclusions as those drawn for Figure 2.26 can be drawn here. The extracted RC model does not contain any information as to which nodes are actually being used as input nodes and output nodes, but allows an accurate simulation of the transmission behavior in each direction.

The second example indicates the validity of the node reduction method for models in which coupling capacitances are included. The circuit that is simulated is shown in Figure 2.27. It consists of a vertical piece of polysilicon interconnect that is crossed by a long horizontal wire of polysilicon and metal. At node ’a’ a falling ramp input voltage is applied and the voltage at node ’c’ is simulated with node ’b’ connected to a 5V/40kohm source. In Figure 2.28, the dotted curve gives the result for the extracted RC model while the solid curve is the result of a simulation for a detailed model. (The detailed model has been obtained for this situation by retaining nodes at the contacts and at the overlap area.)

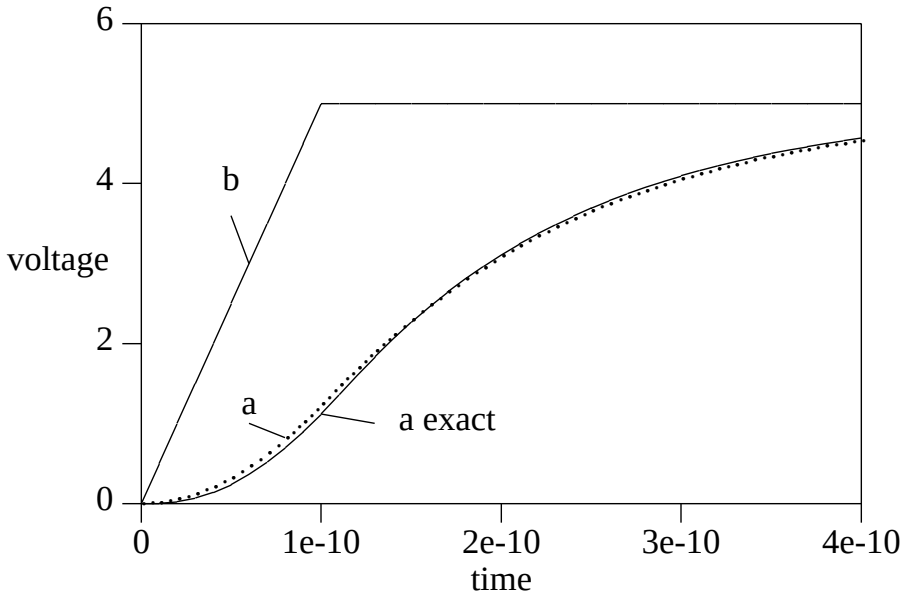


Figure 2.26. Voltage of node ’a’ when ’b’ is used as input.

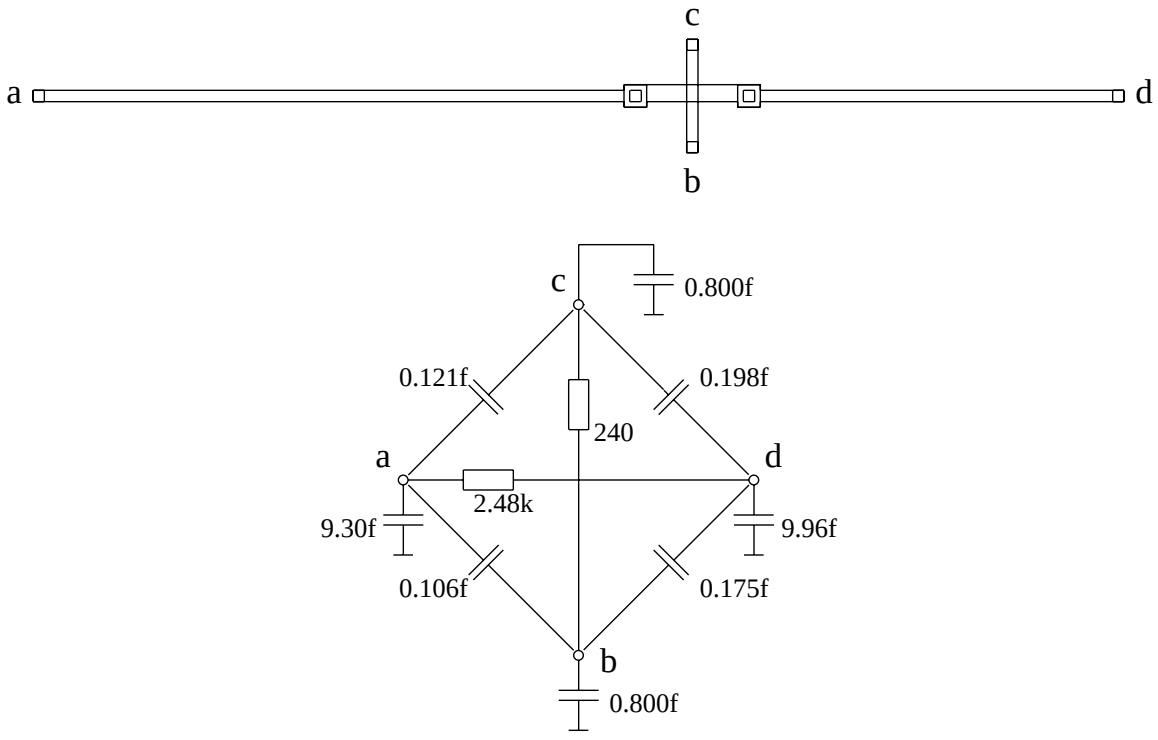


Figure 2.27. Two crossing interconnections and their extracted model.

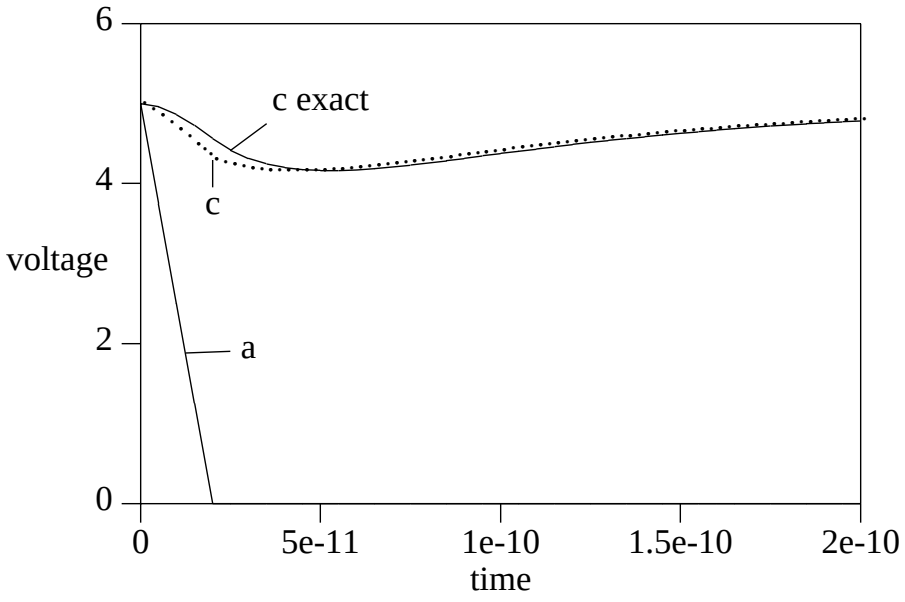


Figure 2.28. Voltage of node 'c' when a ramp input voltage is applied to 'a', and 'b' is connected to a 5V/40kohm source.

2.4 Solution scheme

2.4.1 Introduction

To derive the final RC network from the initial RC network, we have to simplify the set of simultaneous equations (see Section 2.3.4)

$$I(s) = Y(s)\,\Phi(s), \tag{2.112}$$

where $I(s)$ is the vector of the Laplace transforms of the node currents in the initial network, $\Phi(s)$ is the vector of the Laplace transforms of the node potentials in the initial network, and $Y(s)$ is the node admittance matrix of the initial network with entries

$$Y_{ij} = A_{ij} + sB_{ij} \qquad (i = 1 \cdots N,\ j = 1 \cdots N), \tag{2.113}$$

with

$$A_{ij} = -G_{ij} \qquad (i \neq j) \tag{2.114a}$$

$$B_{ij} = -C_{ij} \qquad (i \neq j) \tag{2.114b}$$

$$A_{ii} = \sum_{j=1,\ j\neq i}^{N} G_{ij} \tag{2.114c}$$

$$B_{ii} = C_i + \sum_{j=1,\, j\neq i}^{N} C_{ij}. \tag{2.114d}$$

The solution is obtained by repeatedly using algorithm 2.90 and provides a set of simultaneous equations

$$I'(s) = Y'(s)\,\Phi'(s), \tag{2.115}$$

where $I'(s)$ is the vector of the Laplace transforms of the node currents in the final network, $\Phi'(s)$ is the vector of the Laplace transforms of the node potentials in the final network, and $Y'(s)$ is the node admittance matrix of the final network with entries

$$Y'_{ij} = A'_{ij} + sB'_{ij} \qquad (i = 1 \cdots M,\ j = 1 \cdots M), \tag{2.116}$$

with

$$A'_{ij} = -G'_{ij} \qquad (i \neq j) \tag{2.117a}$$

$$B'_{ij} = -C'_{ij} \qquad (i \neq j) \tag{2.117b}$$

$$A'_{ii} = \sum_{j=1,\, j\neq i}^{N} G'_{ij} \tag{2.117c}$$

$$B'_{ii} = C'_i + \sum_{j=1,\, j\neq i}^{N} C'_{ij}. \tag{2.117d}$$

If the initial matrix $Y(s)$ is a full matrix - i.e. $Y_{ij} \neq 0$ for every pair i, j of $Y(s)$ - the solution of $Y'(s)$ will require

$$\sum_{k=1}^{N-M} \frac{1}{2}(N-k)(N-k+1) = \frac{1}{6}N^3 - \frac{1}{4}N^2 + \frac{1}{12}N - \frac{1}{6}M^3 + \frac{1}{4}M^2 - \frac{1}{12}M \tag{2.118a}$$

$$= O(N^3) \tag{2.118b}$$

operations of the type given by 2.51 or 2.90. The number of different entries that should be stored in this case is equal to $\frac{1}{2}(N-1)N = O(N^2)$. However, in practice $Y(s)$ is sparse - i.e. $Y_{ij} = 0$ for several pairs i, j of $Y(s)$ - since each node in the initial network is directly connected to only a limited number of other nodes via resistances and/or capacitances. This allows us to reduce the number of computations as well as the storage requirements [33]. The amount of reduction that is achieved is dependent on the order in which the nodes are eliminated, i.e. on the numbering scheme of the nodes in the finite-element mesh, and on the matrix storage technique used.

The influence of the numbering scheme on the computation complexity and the storage requirements is illustrated by the example shown in Figure 2.29. For a particular finite-element mesh two different numbering schemes are shown in Figure 2.29a and Figure 2.29b. The lower triangular parts of the corresponding matrices *Y(s)* are shown in Figure 2.29c and Figure 2.29d. Non-zero entries are denoted by an x and fill-ins during the elimination process are denoted by an o. When eliminating nodes $1 \cdots 5$, the numbering scheme used in Figure 2.29a will require $5+5+5+5+2=22$ non-trivial operations, and the numbering scheme given in Figure 2.29b will require $5+9+9+5+2=30$ non-trivial operations. Thus, for the numbering scheme given in Figure 2.29a both the storage requirements and the computation complexity are reduced in comparison with the numbering scheme given in Figure 2.29b.

In the following, we will first describe two techniques that reduce both the computation complexity, and the storage complexity for the derivation of the final RC network from the initial RC network. One technique, the frontal solution technique, reduces computation and storage complexity by providing a simple but efficient enumeration technique for the nodes in the finite-element mesh and "on the fly" construction and solution of the matrix *Y(s)*. The other technique, the graph representation, reduces computation and storage complexity by explicitly storing the information about the sparsity of the system. Secondly, we will describe an implementation of the total RC extraction method in which both these techniques are realized. In this implementation, which is a "scanline based" implementation, all steps of the method are executed as a vertical scanline is swept over the layout from left to right. It will be shown that the method can be implemented such that it has, for the extraction of resistances and ground capacitances, a time complexity that is approximately *O(S)*, where *S* is the size of the layout, and a space complexity that is approximately $O(\sqrt{S})$.

2.4.2 The frontal solution method

In the frontal solution method [34, 7] a line-shaped front is moved over the finite-element mesh and the contribution of each element to the matrix *Y(s)* is added to an intermediate matrix $\hat{Y}$ - which called the frontal matrix - as the front passes the element. When the front has passed all elements that are connected to a node, all non-zero entries in the row/column corresponding to the node are known, and the row/column is eliminated from the matrix $\hat{Y}$ in order to reduce memory requirements. It is easily verified from algorithm 2.90 that it is possible to do so without modifying the final result. This process is repeated until all elements have been passed by the scanline and until all non-terminal nodes of the finite-element mesh have been eliminated. Thus, a solution is obtained while maintaining only

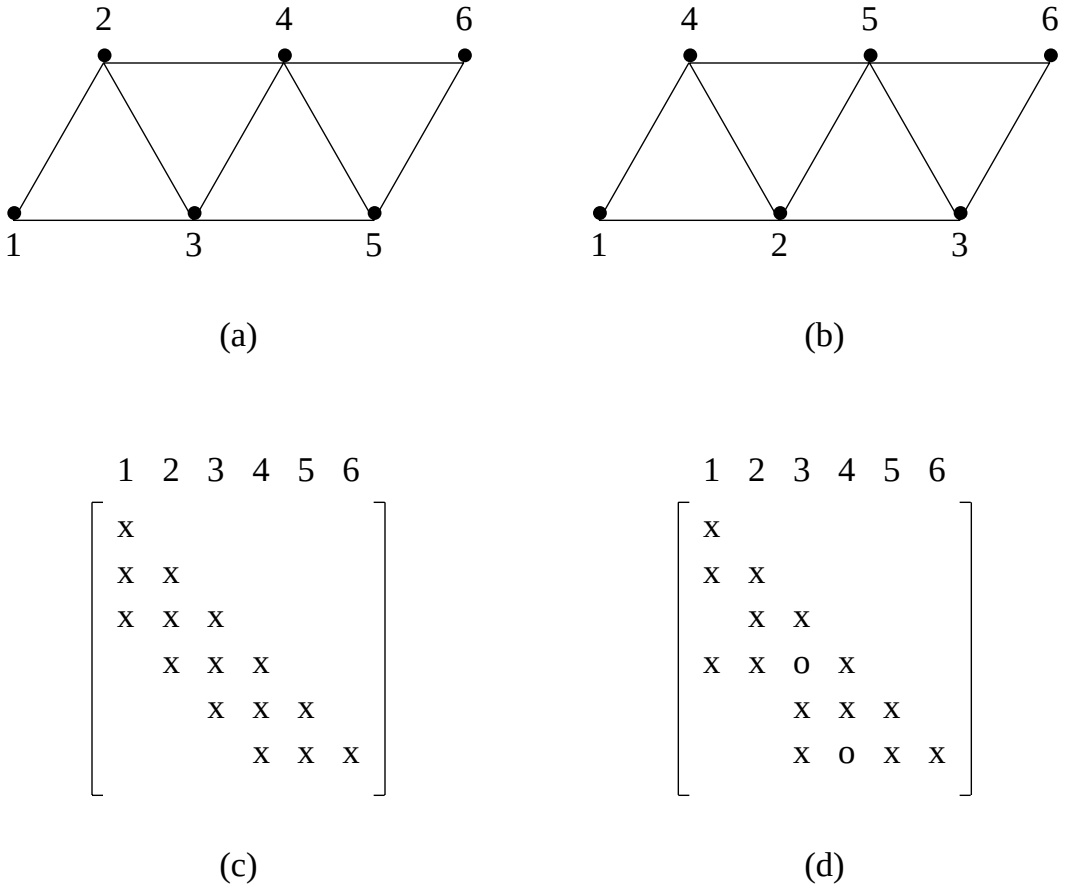


Figure 2.29. Examples of a good numbering scheme and a poor numbering scheme for the solution of a finite-element mesh. The node admittance matrices for the finite-element meshes of (a) and (b) are shown in respectively (c) and (d). Non-zero elements are denoted by an x and zero fill-ins during the elimination process are denoted by an o.

the frontal matrix $\hat{Y}$, which has a maximum dimension that will typically be smaller than the dimension of the total matrix *Y(s)*.

The method is illustrated for the finite-element mesh shown in Figure 2.30a. Node 1 and node 6 are not eliminated since it is assumed that they are terminal nodes. When the front is represented by a vertical line that is swept over the mesh from left to right, the first-element to be visited is *A*. Plugging the conductances of *A* into the front matrix *Y* gives a result which is schematically shown in Figure 2.30b. Next, element *B* is visited, and - when adding its conductances to the previous matrix - the matrix in Figure 2.30c is found. Now, all conductances that are connected to node 2 are known and the node is eliminated in order to reduce the size of the matrix. The above steps are repeated for elements *C* and *D*, while

respectively eliminating after the addition of element *C* node 3 and after the addition of element *D* node 4 and node 5. It is easily verified that during all steps the dimension of the front matrix does not exceed the value 4.

The number of occupied rows and columns in the frontal matrix is determined by the nodes that have been passed by the front and that are connected to the elements that are crossed by the front at a certain time, plus the nodes that have been passed and that are not eliminated since they are terminal nodes. Typically, the maximum number of occupied rows/columns in the frontal matrix will be much smaller than the dimension of the original matrix *Y(s)*. This results in a significant reduction in memory usage as compared to the case where the solution is obtained directly from the matrix *Y(s)*. Also, a sub-optimal numbering scheme is found since the front enumerates the nodes in such a way that nodes that are connected to each other via an edge are relatively close in numbering scheme (compare with a situation where the nodes are numbered in arbitrary order; the numbering scheme of the frontal solution method will give a multiple banded matrix while an arbitrary numbering scheme will give a matrix in which non-zero entries occur at arbitrary positions). The latter assures a relatively low number of fill-ins and, consequently, a relatively low computation complexity.

2.4.3 The graph representation

In the graph representation the matrix is stored as a network consisting of nodes and edges, instead of as a set of rows or columns as used for the explicit matrix representation. Each row/column of the matrix corresponds to a node in the network, and each non-zero entry of the matrix corresponds to an edge in the network. With each edge a value $G_{ij} + sC_{ij}$ is associated, corresponding to the value of the resistor and/or capacitor that is represented by the edge. The use of a graph representation for the solution of the final network provides the following advantages over the use of a matrix representation.

First: for a graph-based representation only the non-zero network elements occupy a memory position, while for a matrix representation also the positions that contain zero’s and that fall within the band of the matrix will increase the memory requirements of the program (we assume that the matrix is stored row- or column-wise so that the size of a row or column corresponds to the bandwidth of the matrix at that position). The number of these positions can be minimized by using a well-chosen numbering scheme, but in most cases they cannot be avoided, and especially for situations with many irregularly shaped interconnections (including their coupling capacitances) their number can become very large.

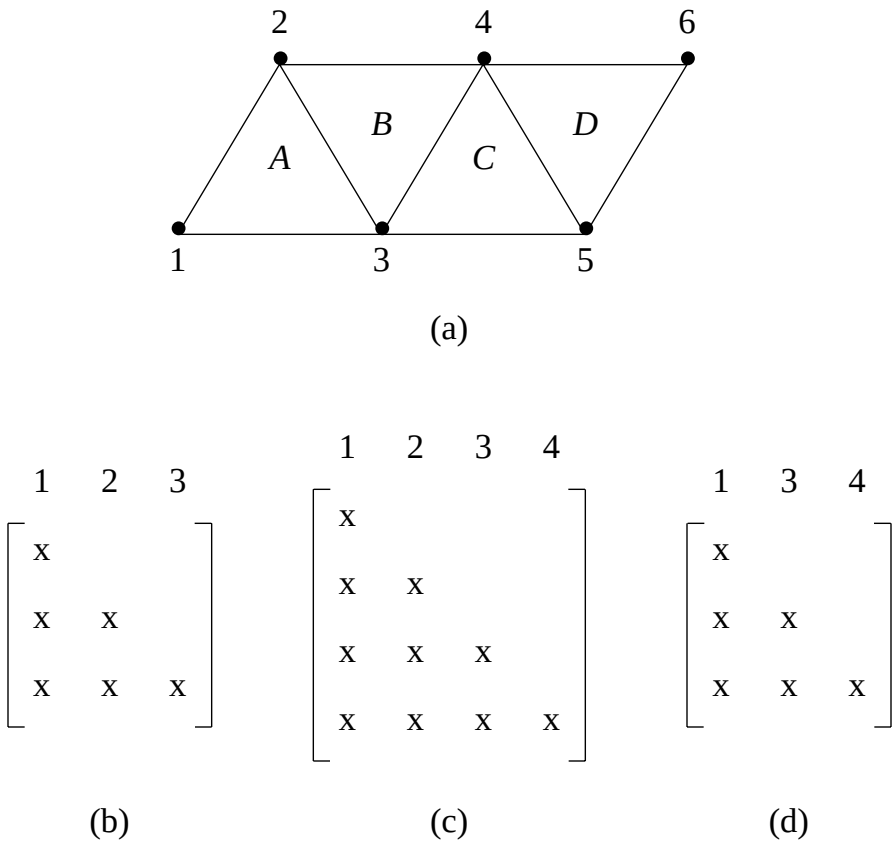


Figure 2.30. Finite-element mesh with elements $A \cdots D$ and nodes $1 \cdots 6$, and the first three occurrences of the front matrix when applying a frontal solution scheme.

Second: graph-based implementations easily allow for operations that are restricted to the neighborhood of the node or element that is affected. For example, when eliminating a node from the network, the elements and nodes that are directly connected to the node are easily found from searching the network, starting at the node that is eliminated, while the other parts of the network are not involved. Also, an element is added to the network without evaluating or updating the parts of the network that are not directly connected to the nodes between which the element is added. Thus, for a graph representation, the time required to perform node and element operations is only dependent on the number of nodes and elements that are directly connected to the node and are independent of the total size of the network. In contrast, for the matrix representation, when a node is eliminated, all row/column positions of the node need to be checked on non-zero entries.

Third: a graph-based representation matches well with the desired form of program-output, and easily allows for graph-based post-processing steps that may

be applied to the extracted network (see also Section 2.5.2).

A disadvantage of the graph-based implementation is that it requires additional memory space to store the connectivity of the network. For some cases (e.g. for dense networks where there are many coupling capacitances) this may increase the memory requirements to a level that exceeds the value of the memory requirements found for the matrix representation.

2.4.4 Scanline implementation

Scanline techniques are applied in VLSI CAD programs that perform geometrical operations such as design-rule checking and layout-to-circuit extraction [35, 36, 37, 38]. In such a program a scanline is moved over the layout, say from left to right, and a cross-section of the layout along the scanline is maintained on which the different operations are performed. In the scanline implementation of the RC modeling method, all steps of the method - construction of the finite-element mesh, the assembling of the initial RC network, and the solution of the final RC network - are executed as the scanline is moved over the layout [16]. The scanline is interpreted as the front of the frontal solution solution method in order to obtain low time and space complexities [7]. The graph representation is used to exploit the sparsity of the RC network that is extracted.

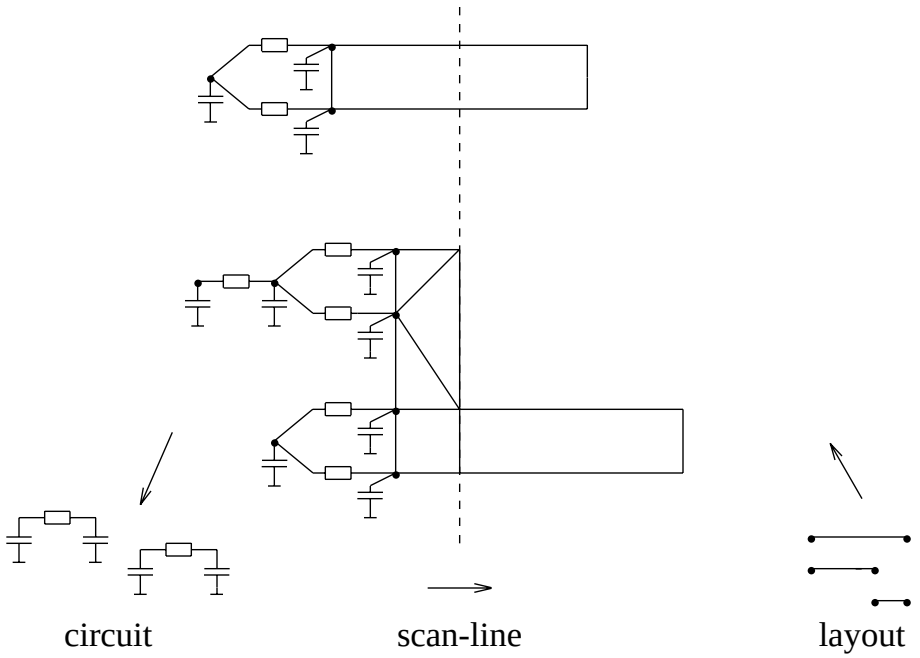


Figure 2.31. Scanline-based implementation of the extraction method.

The scanline-based implementation of the RC modeling method in a layout-to-circuit extraction program is schematically shown in Figure 2.31. The input for the extraction process as depicted in Figure 2.31 consists of non-vertical contour edges that are sorted, first, on their left x coordinate and, second, on their left y coordinate [38]. As the scanline is swept over the layout, the layout is first decomposed into trapezoids that have a unique mask combination and a maximum length. When a trapezoid contains interconnect it is split into boxes and triangles, and the edge resistances and node capacitances are computed. When all resistances and capacitances for a node are found, the node is eliminated in order to reduce memory requirements. Using a perimeter/surface method to compute capacitances, and when only computing ground capacitances, this is the case as soon as the scanline has passed all boxes and triangles of which the node is part of. When the scanline has past the right-most part of the interconnection, the RC network for the interconnection is finished and the result is written to disk. (Note that the above implementation uses only a very simple finite-element mesh. To generate a finer and more accurate finite-element mesh, each trapezoid should be retained in memory a little longer and it should be split into a larger number of boxes and triangles.)

To evaluate the memory requirements of the above implementation, we consider the items that are in core at a particular moment during the extraction process. From the above it follows (see Figure 2.31) that these items are (1) the contour edges that are intersected by the scanline, (2) the nodes that are immediately to the left of the scanline and that are part of the boxes and triangles that are not yet finished because they are intersected by the scanline, (3) the terminal nodes that are to the left of the scanline and that are on a conductor that is intersected by the scanline, and (4) the resistances and capacitances that are present between the nodes that are in core. In the following, we assume that S is the size of the layout (measured in the number of contour edges). The expected number of edges that is intersected by the scanline will be proportional to the length of the layout along the direction of the scanline, which is $O(\sqrt{S})$ [39]. The expected number of interconnections that is intersected by the scanline is found in a similar way also to be $O(\sqrt{S})$. The average number of terminal nodes of an interconnection as well as the average number of resistances for an interconnection that is in core at a particular moment will be determined by the length of that interconnection. When the average interconnection length is independent of the size of the layout, the number of terminals and the number of resistances of an interconnection will be bounded by a constant and the expected space complexity of the method is found to be $O(\sqrt{S})$. If the average interconnection length is dependent on the size of the layout (which is more realistic in many cases) or if coupling capacitances are

extracted, an expected space complexity is found that is worse than $O(\sqrt{S})$.

The computation complexity follows by evaluating the cost to eliminate a node from the network. When N_R is the number of resistances that is connected to a node prior to its elimination, and N_C is the number of capacitances that is connected to it, it follows from 2.110 and 2.111 that the elimination of a node takes $O(N_R N_C + N_R^2)$ time. If we consider a situation where only ground capacitances are extracted and no coupling capacitances are extracted we have $N_C = 1$. The value of N_R is determined by the number of terminal nodes that is to the left of the scanline and that is on the same conductor as the node to be eliminated, plus the number of nodes that are on the same boxes and triangles as the node to be eliminated. If the length of the interconnections is independent of the size of a layout, the number of terminal nodes and the number of resistances for an interconnection will be bounded and the average time to eliminate a node will be approximately constant. Since the number of nodes is proportional to the size of the layout we then arrive at a time complexity that is $O(S)$. However, if the average interconnection length is dependent on the size of the layout or if coupling capacitances are extracted a computation complexity will be found that is more than $O(S)$.

The space and time complexities of the scanline implementation are illustrated by Table 2.7 and Table 2.8, and Figure 2.32 and Figure 2.33. These tables and figures show extraction results that were obtained when extracting ground capacitances and all polysilicon and diffusion resistances for four different circuits. The size of the circuits is indicated by the number of non-vertical contour edges of the circuits. Table 2.7 shows the total number of nodes and the total number of resistances that are generated, as well as the maximum number of nodes and the maximum number of resistances that are in core at a particular moment. From Figure 2.32 is observed that the maximum number of nodes as well as the maximum number of resistances that are in core at a particular moment have a complexity that is approximately $O(\sqrt{S})$. Table 2.8 shows the worst-case total elimination cost and the measured total elimination cost that are found for each of the four different designs. The worst-case total elimination cost is found from 2.118a, while the measured total elimination cost is found by summing the expressions $N_R + N_R^2$ for each node that is eliminated. It is found from Figure 2.33 that the measured elimination cost is approximately $O(S)$.

Table 2.7. Space complexity.

		total		max. in core	
circuit	edges	nodes	res.	nodes	res.
latch	115	508	1,399	92	132
random counter	1,406	6,623	18,139	263	437
memory struct.	10,296	44,562	122,483	1,138	1,984
28 bit counter	27,880	122,311	387,727	1,715	2,422

Table 2.8. Time complexity.

		elimination cost	
circuit	edges	worst case	measured
latch	115	$2.18 \cdot 10^{7}$	$1.52 \cdot 10^{4}$
random counter	1,406	$4.84 \cdot 10^{10}$	$2.08 \cdot 10^{5}$
memory struct.	10,296	$1.47 \cdot 10^{13}$	$2.01 \cdot 10^{6}$
28 bit counter	27,880	$3.05 \cdot 10^{14}$	$7.58 \cdot 10^{6}$

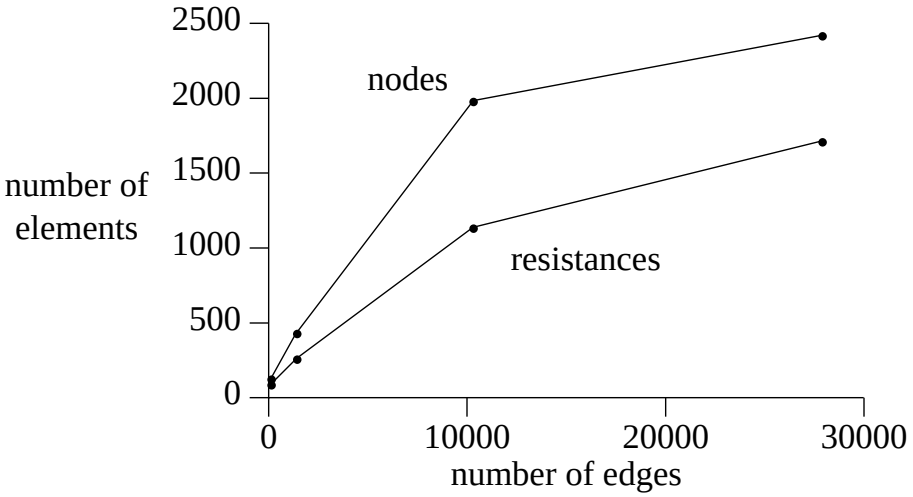


Figure 2.32. Maximum number of nodes and maximum number of resistances that are in core as a function of the size of the circuit.

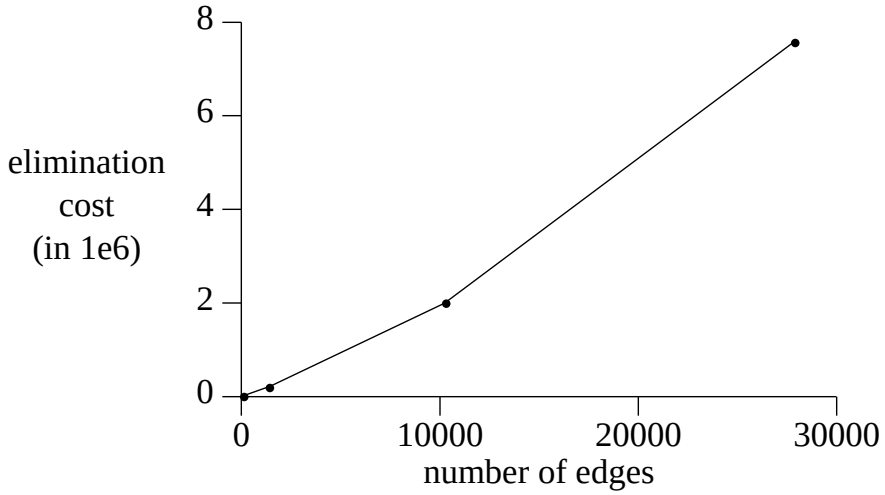


Figure 2.33. Measured elimination cost as a function of the size of the circuit.

2.5 Application to extraction

2.5.1 Introduction

In this section we will discuss the application of the previously described RC network extraction method in a practical layout-to-circuit extraction program. We will first describe how the final RC network that is extracted is adapted to a form that is suitable for the different forms of verification that are performed after the circuit has been extracted. In order to allow efficient verification of the extracted RC network afterwards, the number of nodes and elements in the RC network should be not too high. Therefore we will describe some methods to reduce the number of nodes and elements in the RC models that are extracted (see also [14] and [40] for example). After that, we will discuss the layout circuit extraction program SPACE, in which the RC network extraction method has been implemented.

2.5.2 Practical RC Models

Suppose an interconnection has t terminals. The maximum number of resistances that will be generated for it is equal to the number entries that are stored in the strictly upper triangular part of a matrix of size $t \times t$ which is $\frac{1}{2}(t-1)\,t$. The number of ground capacitances that is generated will be equal to or smaller than t. The number of coupling capacitances for the total network is dependent on the the total size of the network and will be bounded by $\frac{1}{2}(T-1)\,T$, where T is the total number of terminals in the network. In general, the above numbers may be much too high to allow an efficient verification of the network. Therefore, some

heuristics should be applied on the final RC model to reduce the number of nodes, resistances and capacitances. These heuristics should keep the number of nodes, resistances and capacitances of the model at a value that allows efficient verification of the final model, without giving inaccurate results. In the following some of these heuristics are presented. They are explained by using the interconnection example shown in Figure 2.34 and Figure 2.35.

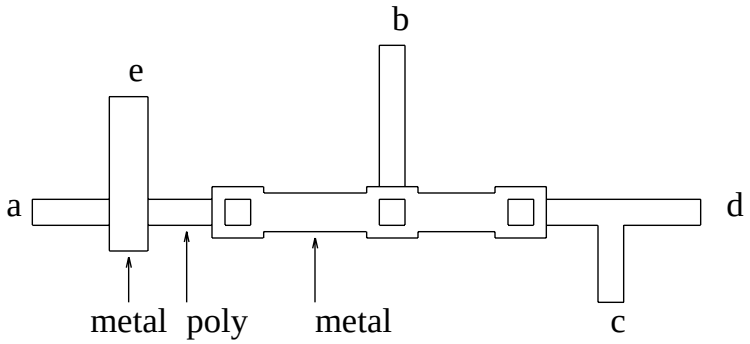


Figure 2.34. Interconnection example.

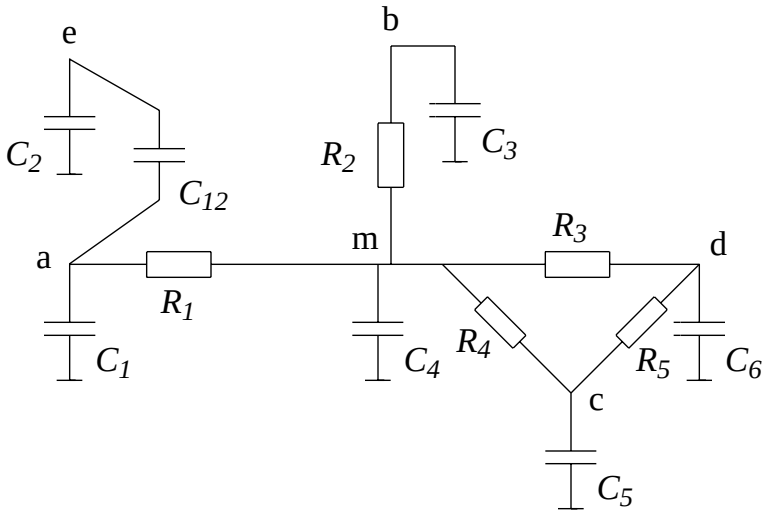


Figure 2.35. RC model for the interconnection example shown in Figure 2.34.

2.5.2.1 Retainment of non-terminal nodes

One way to reduce the number of elements in the RC network that is extracted is by retaining non-terminal nodes that have an articulation degree > 1. The articulation degree of a node is defined as the number of pieces in which the resistance graph would break if the node and its connected resistances were removed. For example, the articulation degree of node *m* in Figure 2.35 is 3. Nodes in the initial network that have an articulation degree > 1 will automatically

be introduced by interconnection areas that are (approximately) equi-potential areas such as metal wires (node *m* in Figure 2.34). They may further artificially be introduced by splitting the interconnections along (approximate) equi-potential lines. As an example of the retainment of a node with an articulation degree > 1, we consider node *m* in Figure 2.35. The number of resistances in the network shown in Figure 2.35 is equal to 5. If node *m* is eliminated the number of resistances in the network will become 6. Thus - in this case - it is better not to eliminate node *m*. On the penalty of an extra node, the retainment of a node *i* with an articulation degree *d* will reduce the number of resistances from $\frac{1}{2}(t-1)\,t$ to $\sum_{j=1}^{j=d} \frac{1}{2}(t_{i,j}-1)\,t_{i,j}$, where $t_{i,j}$ is the number of terminals that are part of a branch *j* ($j = 1 \cdots d$) that is connected to node *i*.

2.5.2.2 Removal of large resistances

Another way to reduce the number of elements is by removing resistances that are short-circuited by a resistance path of which the total resistance is much smaller than the value of that resistance. As an example we consider resistor R_3 in Figure 2.35. If $(R_4 + R_5)/(R_3 + R_4 + R_5) < \varepsilon_R$, where ε_R is the maximum error in the total resistance between two terminals in the RC network, R_3 is simply removed.

2.5.2.3 Coalescing of terminal nodes

Coalescing of different terminals of one interconnection is especially useful for wires with many transistor connections such as, for example, the interconnection wires in RAMs and PLAs. In this situation, the number of nodes and elements in the final model can be reduced by eliminating one terminal and adding the transistor connections and name labels of that terminal to the other terminal. The coalescing is only executed, say, when the resistance between the terminals is less than a particular threshold value $R_{\min}$. As an example, we consider the resistance network in Figure 2.36a, which has 9 gate connections and 8 resistances. By recursively executing the coalescing of terminal nodes that are separated by only a small resistance, a simplified model may be found as shown in Figure 2.36b. This reduced network has only 3 nodes and 2 resistances while its behavior may approximately be equal to the behavior of the network as shown in Figure 2.36a.

2.5.2.4 Splitting of small coupling capacitances

The number of coupling capacitances in the final RC model is often drastically reduced by reconnecting small coupling capacitances to ground. Since the influence of a coupling capacitance on the behavior of the network is to a large extent determined by the ratios between the coupling capacitance and the ground

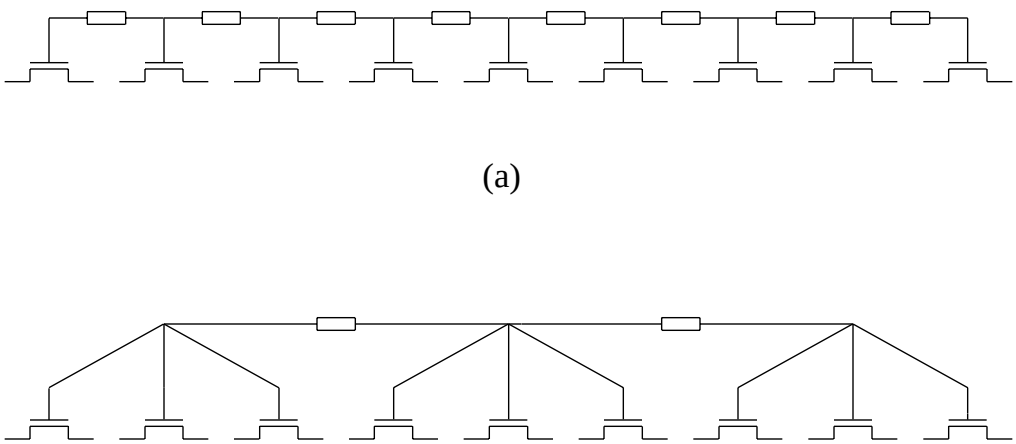


Figure 2.36. Resistance network of a wordline in a RAM circuit before (a) and after (b) coalescing of terminal nodes.

capacitances that are connected to it, it is advantageous to judge the removal of a coupling capacitance by comparing its value with the values of the ground capacitances that are connected to the coupling capacitance. If the value of the coupling capacitance is significantly smaller than the values of the ground capacitances, the coupling capacitance is removed and its value is added to the value of both ground capacitances in order to ensure that the short-circuit capacitance of each node is unchanged. As an example we consider coupling capacitance C_{12} between node e and node a. When $C_{12}/C_1 < F_C$ and $C_{12}/C_2 < F_C$, where F_C is maximum ratio between a coupling capacitance and its ground capacitances, C_{12} is removed and its value is added to both C_1 and C_2.

2.5.3 The SPACE layout to circuit extractor

The extraction method has been implemented in a layout to circuit extraction program called SPACE (= Submicron Parasitic Artwork to Circuit Extractor) [41]. The program SPACE is part of the Nelsis VLSI design system [42], which consists of a framework plus a set of tools for the design and verification of digital and analog integrated circuits [43]. The task of SPACE is to find the equivalent circuit representation of a layout description, including its transistors, its connectivity and its interconnect resistances and capacitances (see Figure 2.37). The result is used by verification programs such as a network comparison program [44] and different types of simulators [1, 45] to check and validate the performance of the circuit that has been implemented. If the behavior of the circuit is not within its specifications, the layout is modified and another extraction and verification run

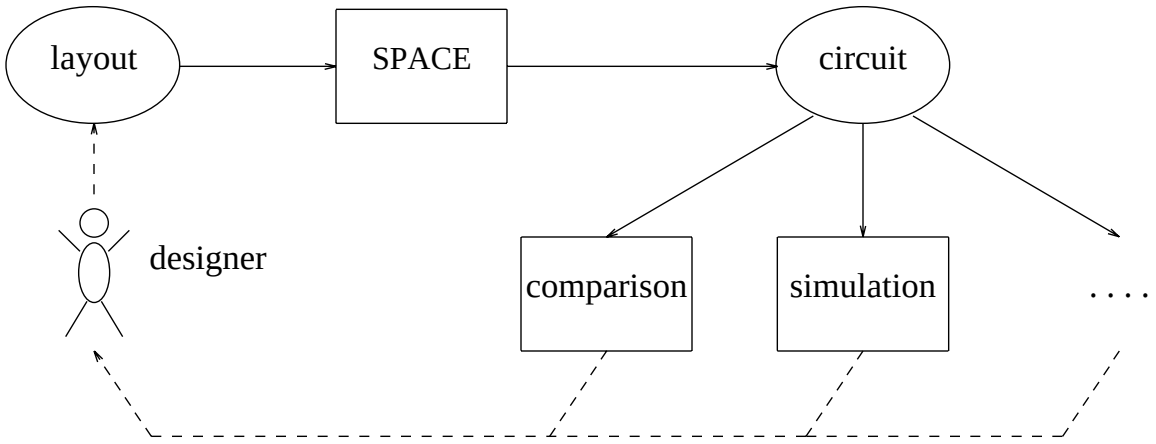


Figure 2.37. The program SPACE in a design environment.

are executed, until the designer is satisfied with the result.

The program SPACE is capable of extracting MOS circuits and bipolar circuits. It can extract the resistances, ground capacitances and coupling capacitances of interconnections. Both cross-over coupling capacitances and lateral coupling capacitances are recognized. In a user-defined element definition file, it is specified how the various elements are recognized from the different combinations of mask layers. For example, an n-MOS transistor is recognized when a diffusion mask is overlapped by a polysilicon mask, while no contact mask is present. The layout that is extracted is described in a layout language like e.g. CIF or GDSII, [46] or it is, for example, generated by a layout editor program [47]. Before the actual extraction takes places, the layout is first converted into a set of non-vertical line segments. During this step it is decided whether SPACE will generate a hierarchical or flat circuit description. Some network reduction heuristics are applied on the final RC models that are generated for the interconnections, in order to reduce the number of nodes and elements of the models. The output is a netlist in database format, which may converted to a circuit language such as SPICE [48] or EDIF [49].

Table 2.9, Table 2.10 and Table 2.11 show some extraction results that were obtained for four different digital MOS circuits, using SPACE. All extraction times that are given were obtained on a Sun SPARC 1+ workstation. The results in Table 2.9 refer to a case where only the transistor elements and the connectivity of the circuit were extracted. The number of transistors and the number of nodes that are found for each circuit are shown as well the total extraction time. Table 2.10 shows extraction results for the case where ground capacitances and all polysilicon and diffusion resistances were extracted. In this table also the number

of resistances and the number of capacitances that are generated are given. The values of the parameters that were used for the application of the network reduction heuristics are as follows: Minimum articulation degree for non-terminal nodes that are retained in the final model = 3; $\varepsilon_r = 0.01$; and $R_{\min} = 100$. Finally, Table 2.11 shows extraction results for the case where also coupling capacitances were extracted. In addition to the previous values for the network reduction heuristics $F_C = 0.05$ was used here.

Table 2.9. Extraction results on a Sun SPARC 1+ workstation for four different MOS circuits when only extracting transistor elements and connectivity.

circuit	transistors	nodes	time (sec.)
latch	11	11	0.4
random counter	149	101	1.5
memory struct.	821	483	8.8
28 bit counter	1176	403	24.5

Table 2.10. Extraction results on a Sun SPARC 1+ workstation when extracting ground capacitances and polysilicon and diffusion resistances.

circuit	transistors	nodes	cap.	res.	time (sec.)
latch	11	27	26	20	1.0
random counter	149	287	286	260	9.8
memory struct.	821	1753	1752	1964	96.6
28 bit counter	1176	2149	2148	2064	227.0

Table 2.11. Extraction results on a Sun SPARC 1+ workstation when extracting ground and coupling capacitances as well as polysilicon and diffusion resistances.

circuit	transistors	nodes	cap.	res.	time (sec.)
latch	11	27	60	20	1.2
random counter	149	287	624	260	13.7
memory struct.	821	1753	5062	1964	222.5
28 bit counter	1176	2149	5006	2064	334.8

References

1. L.W. Nagel, ‘‘SPICE2: A computer program to simulate semi-conductor circuits,’’ ERL Memo. ERL-M520, Electronics Res. Lab., Univ. California, Berkeley (1975).

2. P.M. Hall, ‘‘Resistance calculations for thin film patterns,’’ *Thin Solid Films* **1** pp. 277-295 (1967/68).

3. C.M. Sakkas, ‘‘Potential distribution of multi-terminal DC resistance computations for LSI technology,’’ *IBM J. Res. Develop.* **23**(6) pp. 640-651 (Nov. 1979).

4. M. Horowitz and R.W. Dutton, ‘‘Resistance extraction from mask layout data,’’ *IEEE Trans. on CAD* **CAD-2**(3) pp. 145-150 (July 1983).

5. Erich Barke, ‘‘Resistance calculations from mask artwork data by finite element method,’’ *Proc. 22nd Design Automation Conference,* pp. 305-311 (1985).

6. S.P. McCormick, ‘‘EXCL: A circuit extractor for IC designs,’’ *Proc. 21st Design Automation Conference,* pp. 616-623 (1984).

7. P. Kazil and P. Dewilde, ‘‘A Simple and Fast Method for Obtaining Resistance of VLSI Interconnect,’’ *Proc. IEEE ICCD-86,* pp. 342-345 (Oct. 1986).

8. Takashi Mitsuhashi and Kenji Yoshida, ‘‘A Resistance Calculation Algorithm and its Application to Circuit Extraction,’’ *IEEE Trans. CAD* **CAD-6**(3) pp. 337-345 (May 1987).

9. J.E. Hall, D.E Hocevar, P. Yang, and M.M. McGraw, ‘‘SPIDER - A CAD System for Modeling VLSI Metallization Patterns,’’ *IEEE Trans. CAD* **CAD-6**(6) pp. 1023-1031 (Nov. 1987).

10. B.R. Chawla and H.K. Gummel, ‘‘A Boundary Technique for Calculation of Distributed Resistance,’’ *IEEE Trans. on Electron Devices* **ED-17** pp. 915-925 (Oct. 1970).

11. A.C. van der Woerd, J.P.M. van Lammeren, R.J.H. Janse, and R.H. van Beynhem, ‘‘Calculation of the Resistance Value of Laser-Trimmable Planar Resistors in an Interactive Mask-Layout Design System,’’ *IEEE Journal of Solid-State Circuits* **SC-19**(4) (August, 1984).

12. T. Sakurai, ‘‘Approximation of wiring delay in MOSFET LSI,’’ *IEEE Journal of Solid-State Circuits* **SC-18**(4) pp. 418-426 (Aug. 1983).

13. N.P. van der Meijs and J.T. Fokkema, ‘‘VLSI circuit reconstruction from mask topology,’’ *INTEGRATION, the VLSI Journal* **2**(2) pp. 85-119 (1984).

14. S.-L. Su, V.R. Rao, and T.N. Trick, ‘‘A Simple and Accurate Node Reduction Techique for Interconnect Modeling in Circuit Extraction,’’ *Proc. IEEE ICCAD-86*, pp. 270-273 (Nov. 1986).

15. M.Glez Harbour and J.M. Drake, ‘‘Simple RC model for integrated multiterminal interconnections,’’ *IEE Proceedings* **135**(1) pp. 19-23 (Feb. 1988).

16. A.J. van Genderen and N.P. van der Meijs, ‘‘Extracting Simple but Accurate RC Models for VLSI Interconnect,’’ *Proc. ISCAS-88*, Helsinki, Finland, pp. 2351-2354 (June 7-9, 1988).

17. J.D Bastian, M. Ellement, P.J. Fowler, C.E. Huang, and L.P. McNamee, ‘‘Symbolic parasitic extractor for circuit simulation (SPECS),’’ *Proc. 20st Design Automation Conference*, pp. 346-352 (1983).

18. D. Stark and M. Horowitz, ‘‘REDS: Resistance extraction for digital simulation,’’ *Proc. 24th Design Automation Conference*, pp. 570-573 (1987).

19. G.E. Forsythe and W.R. Wasow, *Finite-Difference Methods for Partial Differential Equations*, Wiley, New York (1960).

20. M.Glez Harbour and J.M. Drake, ‘‘Calculation of Multiterminal Resistances in Integrated Circuits,’’ *IEEE Trans. on Circuits and Systems* **CAS-33**(4) pp. 462-465 (April 1986).

21. J.K. Binns and P.J. Lawrenson, *Analysis and Computation of Electric and Magnetic Field Problems*, Pergamon, New York (1973).

22. A. George and J.W. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, N.J. (1981).

23. P.P. Silvester and R.L. Ferrari, *Finite Elements for Electrical Engineers*, Cambridge University Press (1983).

24. O.C. Zienkiewicz and K. Morgan, *Finite Elements and Approximations*, John Wiley & Sons (1983).

25. D.H. Norrie and G. de Vries, *The Finite Element Method,* Academic Press, New York (1973).

26. F.R. Gantmacher, *The Theory of Matrices,* Chelsea Publishing Company, New York (1959).

27. W.C. Elmore, ‘‘The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers,’’ *J. Applied Physics* **19** pp. 55-63 (Jan. 1948).

28. M.Glez. Harbour and J.M. Drake, ‘‘Calculation of Signal Delay in Integrated Interconnections,’’ *IEEE Trans. on Circuits and Systems* **CAS-36**(2) pp. 272-276 (Feb. 1989).

29. N. Ahmed and T. Natarajan, *Discrete-Time Signals and Systems,* Reston Publishing Company, Reston, Virginia (1983).

30. T.M. Lin and C.A. Mead, ‘‘Signal Delay in General RC Networks,’’ *IEEE Trans. on Computer Aided Design* **CAD-3**(4) pp. 331-349 (Oct. 1984).

31. J. Rubinstein, P. Penfield, and M.A. Horowitz, ‘‘Signal delay in RC tree networks,’’ *IEEE Trans. on CAD* **CAD-2**(3) pp. 202-211 (July 1983).

32. J.L. Wyatt, ‘‘Signal Delay in RC Mesh Networks,’’ *IEEE Trans. on Circuits and Systems* **CAS-32**(5) pp. 507-510 (May 1985).

33. I.S. Duff, A.M. Erisman, and J.K. Reid, *Direct Methods for Sparse Matrices,* Oxford University Press, Oxford (1986).

34. B.M. Irons, ‘‘A Frontal Solution Program for Finite Element Analysis,’’ *Int. Journal for Numerical Methods in Engineering* **2** pp. 5-32 (1970).

35. J.L. Bentley and T.A. Ottman, ‘‘Algorithms for reporting and counting geometric intersections,’’ *IEEE Trans. on Computers* **C-28**(9) pp. 643-647 (Sept. 1979).

36. U. Lauther, ‘‘An O(N log N) algorithm for boolean mask operations,’’ *Proc. 18th Design Automation Conference,* pp. 555-562 (1981).

37. J. Nievergelt and F.P. Preparata, ‘‘Plane-sweep algorithms for intersecting geometric figures,’’ *Comm. of the ACM* **25**(10) pp. 739-747 (Oct. 1982).

38. N.P. van der Meijs and A.J. van Genderen, ‘‘An Efficient Algorithm for Analysis of Non-Orthogonal Layout,’’ *Proc. ISCAS-89,* pp. 47-52 (May 1989).

39. T.G. Szymanski and C.J. Van Wyk, ‘‘Space efficient algorithms for VLSI artwork analysis,’’ *Proc. 20st Design Automation Conference*, pp. 734-739 (1983).

40. P. Vanoostende, P. Six, and H.J. de Man, ‘‘DARSI: RC Data Reduction,’’ *IEEE Trans. on Computer Aided Design* **10**(4) pp. 493-500 (April 1991).

41. N.P. van der Meijs and A.J. van Genderen, ‘‘Space: An Accurate and Efficient Extractor for Submicron Integrated Circuits,’’ *Delft Progr. Rep.* **12** Delft, pp. 260-279 (1988).

42. P. Dewilde, ed., *The integrated circuit design book: Papers on VLSI design methodology from the ICD-NELSIS Project*, Delft University Press, Delft, the Netherlands (1986).

43. P. van der Wolf, P. Bingley, and P. Dewilde, ‘‘On the Architecture of a CAD Framework: The NELSIS Approach,’’ *Proc. IEEE 1st European Design Automation Conference*, (1990).

44. T. Vogel, ‘‘Connectivity verification based on network comparison,’’ M.S. Thesis, Delft University of Technology, Network Theory Section, Delft, the Netherlands (July, 1986).

45. A.J. van Genderen and A.C. de Graaf, ‘‘SLS: A Switch-Level Timing Simulator,’’ pp. 2.93-2.146 in *The Integrated Circuit Design Book*, ed. P. Dewilde, Delft University Press, Delft, the Netherlands (1986).

46. S.M. Rubin, *Computer Aids for VLSI Design*, Addison-Wesley, Reading, Massachusetts (1987).

47. P. van der Wolf and J. Liedorp, *DALI: User’s Manual*, Delft University of Technology, Network Theory Section, Delft ().

48. A. Vladimirescu, K. Zhang, A.R. Newton, D.O. Pederson, and A. Sangiovanni-Vincentelli, *SPICE version 2G user’s guide*, Univ. of California, Berkeley (Aug. 10, 1981).

49. EDIF Steering Committee, *Electronic Design Interchange Format, Version 2 0 0, Reference Manual*, Electronic Industries Association (1987).

3. 3-D CAPACITANCE EXTRACTION

3.1 Introduction

With the advance in the technology of integrated circuits, the horizontal dimensions of the circuit - i.e. the width of the conductors and the separation between the conductors - are scaled down in order to increase the number of components in the circuit. At the same time, the vertical dimensions - i.e. the thickness of the conductors and their height above the substrate - are left almost unchanged in order to prevent too much increase in interconnect resistance and too high a value of the current density. Thus, the influence of the lateral coupling capacitances between the conductors, as compared to the values of the ground capacitances, increases (see for example Figure 3.1 and Figure 3.2). Therefore, simple approximation formulas like the parallel plate formula are no longer valid to estimate interconnect capacitances, but accurate numerical techniques that solve the Poisson equation in two or three dimensions are required to characterize the values of the interconnect capacitances.

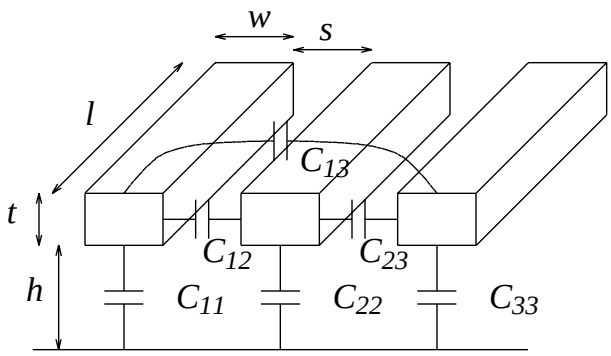


Figure 3.1. Example of three conductors above substrate.

Numerical techniques for the calculation of capacitances for arbitrary conductor configurations (two- and three-dimensional) have been described in, e.g [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]. These methods can be classified in methods that solve the Poisson equation directly: the finite-difference method [5, 6, 8, 11, 12] and the finite-element method [7, 9]; and methods that solve the Poisson equation in integral form: the boundary-element method [1, 2, 3, 4, 10, 13]. Since we consider two- and three-dimensional problems in which many

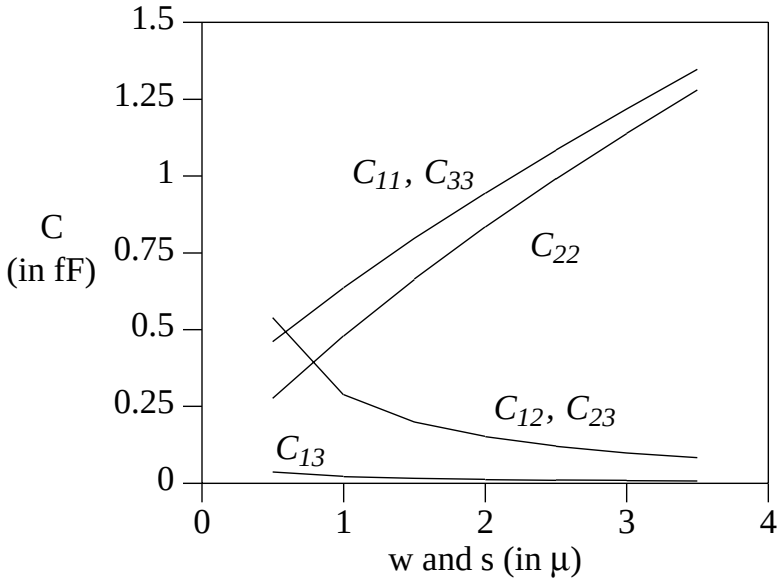


Figure 3.2. Capacitances versus conductor width and spacing for the conductor configuration in Figure 3.1 ($t = 0.5\mu$, $h = 1.5\mu$ and $l = 10\mu$).

conductors may be present (e.g. a static RAM cell contains 7-10 different conductors) the application of the above numerical techniques will in general be very computation and memory intensive. At the same time, useful results should be provided in only a short amount of time, so that the designer can interactively verify and modify his design. Therefore, the capacitance extraction method should be efficient with respect to computation time and memory usage. Further, in order not to blur the final capacitance model with too much detail, the method should be capable of providing a reduced capacitance model in which small capacitances, which are present between conductors that are far apart, are omitted.

The contents of this chapter are as follows: In Section 3.2 a brief overview is presented of the different numerical methods that can be used for capacitance calculation, and the boundary-element method is described in more detail. The boundary-element method uses a grid to model the charge on the conductor surfaces, while the finite-difference method and the finite-element method use a grid to model the electrical field between the conductors. Hence, the complexity of the grid that is used for the boundary-element method is - for a similar problem - significantly lower than the complexity of the grid that is used for the the finite-difference method or the finite-element method.

When applying the boundary-element method, an elastance matrix of size $N \times N$ - where N is the number of grid points - has to be inverted. This matrix is in principle a full matrix and a complete inversion of it will require $O(N^3)$ time and

$O(N^2)$ memory. However, many entries in the inverse of the elastance matrix will be small since they correspond to capacitances between elements that are far apart. Therefore, Section 3.3 describes a matrix inversion technique that computes only an approximate inverse for the elastance matrix that is found for a three-dimensional (3-D) conductor configuration. When using this matrix inversion technique to solve the boundary-element model described in Section 3.2, the computation complexity and the space complexity of the method are reduced, as well as the complexity of the final capacitance model.

In Section 3.4, a solution scheme for the 3-D capacitance modeling method and its approximative matrix inversion technique is described. How parallelization can be used to speed up the computation time of the matrix inversion algorithm, and how an incremental solution lowers the memory requirements of the algorithm is shown. Moreover, it is shown that by subdividing the circuit into vertical (or horizontal) strips, and by using a simple and straight-forward numbering scheme, the capacitance extraction method can be implemented to have a computation complexity that is $O(S)$, where S is the size of circuit, and a space complexity that is constant.

Finally, in Section 3.5, the incorporation of the 3-D capacitance extraction method in a practical layout to circuit extraction program is discussed. This section also presents some extraction results for a practical circuit design.

3.2 Capacitance model

3.2.1 Introduction

An example of a cross-section of an integrated circuit is shown schematically in Figure 3.3. The interconnections (conductors) are located in one or more different horizontal layers, each consisting of an isolating material like silicon oxide (SiO_2) or silicon nitride (Si_3N_4). These layers have been deposited on top of a substrate, which is either a conductor (silicon) or an insulator (e.g. sapphire). For the purpose of capacitance calculation, the interconnections are approximated by ideal conductors, and thus the charge of the conductors is present only on the surfaces of the conductors.

To find the capacitances of the interconnections, a relation has to be derived between the conductor potentials and the conductor charges. This relation is governed by the Poisson equation which states that the sum of the partial second-order derivatives of the potential for any point in the domain in which the conductors are located is proportional to the charge density at that point. For positions where no free charge is present, the Poisson equation reduces to a

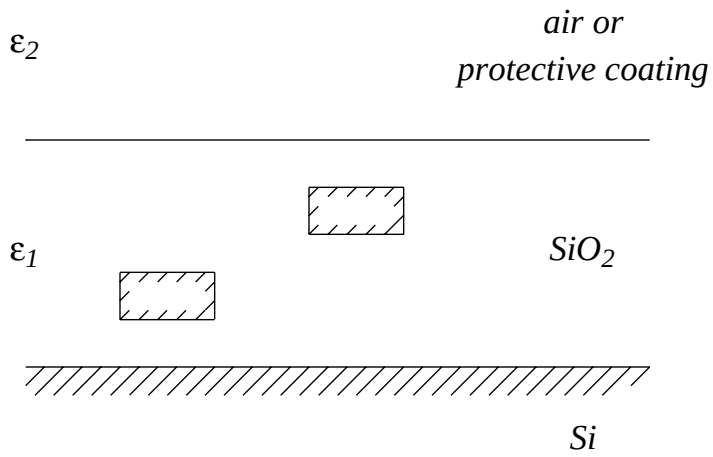


Figure 3.3. Example of a cross-section of an integrated circuit (conductors are hatched).

simpler form, where the sum of the partial second-order derivatives of the potential is equal to zero, which is called the Laplace equation.

In the finite-difference method [5, 6, 8, 11, 12] the Laplace equation is solved by a discretization of it on a two- or three-dimensional rectangular grid that is defined on the area between the conductors. This yields a solution for the potential field between the conductors, from which the conductor charges are found by computing the potential fluxes around the conductors. The capacitances are then obtained from the quotients of the conductor charges and the conductor potentials. To find all capacitances that are connected to one conductor, the finite-difference method solves a system of N linear equations with N variables, where N is the number of grid points. Since the finite-difference method discretizes the potential field between the conductors, the method is well suited to model possible irregularities that may occur in the dielectric between the conductors. However, the number of grid points to model the potential field may reach a very high value for problems in which many conductors are involved. Although the set of equations from which the capacitances are computed is sparse, and it can efficiently be solved by using an iterative technique like e.g. the Incomplete Choleski Conjugate Gradient (ICCG) method [12], the large number of grid points may cause long computation times and a large memory usage, especially for three-dimensional conductor configurations.

In the finite-element method [7, 9] also the potential field between the conductors is modeled, but the space between the conductors is divided into triangular elements (for two-dimensional problems) or prismatic elements (for three-dimensional problems). The potential field is then represented by first-order or

higher-order shape functions that are defined on these elements, and the method of variational calculus [14] is used to obtain an "optimal" (i.e. minimum energy) approximation for the potential field. In a way similar to the finite-difference method, the determination of all capacitances of one conductor requires the solution of a system of N linear equations with N variables, where N is the number of different shape functions or mesh nodes. Because of the use of triangular and prismatic elements, the finite-element method is somewhat more suited to model non-orthogonal boundaries that may occur in conductor configurations. With respect to computations and memory usage, the same considerations are valid for the finite-element method as are valid for the finite-difference method.

The boundary-element method or Green's function method [1, 2, 3, 4, 10, 13] computes a relation between the conductor potentials and the conductor charges by solving the Poisson equation in integral form. First, using the Green's function that is appropriate for the medium in which the conductors are located, the potential at each point in the medium is expressed as an integral over the charge density in the medium. Next, the charge distribution on the surfaces of the conductors is modeled by zero-th order, first-order or higher-order shape functions that are defined on rectangular elements and triangular elements placed on the surfaces of the conductors. Finally, the differences between the conductor potentials and the values of the integral expression are minimized by using some kind of weight functions, and the relation between the conductor potentials and the conductor charges is solved from the resulting set of simultaneous equations. Because the structure of the dielectric is implicitly modeled by the Green's function, the use of the boundary-element method is cumbersome for irregular dielectric structures. However, the complexity of the mesh that is used for the boundary-element method is significantly lower than the mesh as used for the finite-difference method or the finite-element method, since the mesh that is used for the boundary-element method is only created on the surfaces of the conductors. Although the capacitances are obtained from the set of simultaneous equations by inverting a matrix that is in principle a full matrix, the latter will become advantageous when using the approximate matrix inversion technique described in Section 3.3.

In the next section, we will give a description of the boundary-element method for capacitance calculation. For a more extensive description, see for example [15] or [16].

3.2.2 The boundary-element method

Consider a system of M different conductors in a three-dimensional domain V that has a permittivity ε and that is bounded by a contour C (see Figure 3.4). The potential at infinity is chosen as the zero reference potential. The conductors may be accompanied by a ground plane, in which case the potential of this plane is also chosen to be at a zero potential. Let $Q_1 \cdots Q_M$ denote the different charges on each of the conductors and let $\Phi_1 \cdots \Phi_M$ denote the potentials of each of the conductors. To determine the capacitances for the system of M conductors, a relation has to be found between the conductor potentials and the conductor charges resulting in the set of linear equations,

$$\begin{aligned}
Q_1 &= C_{11}\Phi_1 + C_{12}(\Phi_1 - \Phi_2) + \cdots + C_{1M}(\Phi_1 - \Phi_M) \\
Q_2 &= C_{21}(\Phi_2 - \Phi_1) + C_{22}\Phi_2 + \cdots + C_{2M}(\Phi_2 - \Phi_M) \\
&\cdots \\
Q_M &= C_{M1}(\Phi_M - \Phi_1) + C_{M2}(\Phi_M - \Phi_2) + \cdots + C_{MM}\Phi_M.
\end{aligned} \tag{3.1}$$

In Equation 3.1, C_{ij} is the capacitance between conductor i and j and C_{ii} is the ground capacitance of conductor i. Equation 3.1 is sometimes rewritten as

$$\begin{bmatrix} Q_1 \\ Q_2 \\ \cdot \\ Q_M \end{bmatrix} = \begin{bmatrix} C_{s11} & C_{s12} & \cdots & C_{s1M} \\ C_{s21} & C_{s22} & \cdots & C_{s2M} \\ \cdot & \cdot & \cdots & \cdot \\ C_{sM1} & C_{sM2} & \cdots & C_{sMM} \end{bmatrix} \begin{bmatrix} \Phi_1 \\ \Phi_2 \\ \cdot \\ \Phi_M \end{bmatrix}, \tag{3.2}$$

where the matrix C_s, consisting of the entries C_{sij} ($i = 1 \cdots M$, $j = 1 \cdots M$), is called the short-circuit capacitance matrix of the conductor system. By comparison of 3.2 with 3.1 it is easily found that

$$C_{sii} = \sum_{j=1}^{M} C_{ij}, \tag{3.3}$$

and

$$C_{sij} = -C_{ij} \qquad (i \neq j). \tag{3.4}$$

For any point (x, y, z) in the domain V, the electric field E at that point is found from the potential ϕ as

$$E = -\nabla\phi, \tag{3.5}$$

where $\nabla = \{\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\}$. According to Gauss' Law in differential form [17]

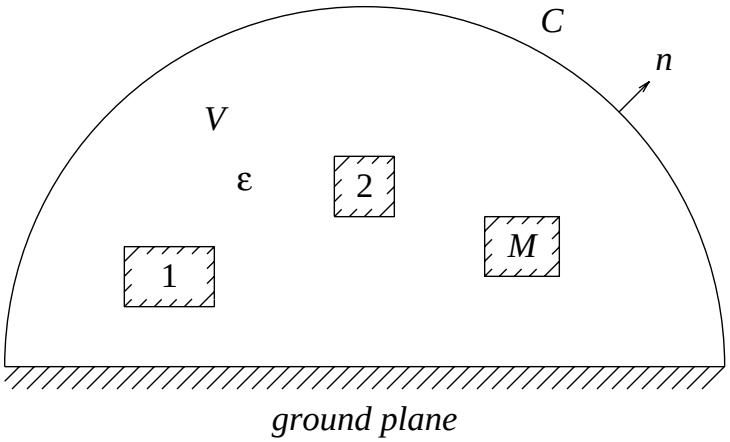


Figure 3.4. A set of *M* different conductors in a domain *V* that has a permittivity ε and a boundary *C*.

$$\nabla \cdot \varepsilon E = \rho, \tag{3.6}$$

where ρ is the charge density at point *(x, y, z)*. When inserting 3.5 into 3.6 we find

$$\nabla \cdot (\varepsilon \nabla \phi) = -\rho, \tag{3.7}$$

which is the relation between the potential and the charge density in *V*. For a separate homogeneous region (where ε is constant) Equation 3.7 can be written as

$$\nabla^2 \phi = -\frac{\rho}{\varepsilon}, \tag{3.8}$$

where $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$, which is known as the Poisson equation. If no charge is present 3.8 reduces to

$$\nabla^2 \phi = 0, \tag{3.9}$$

which is called the Laplace equation.

The boundary conditions for 3.7 (or 3.8 or 3.9) are given by

$$\phi = \Phi_i, \tag{3.10}$$

for any point *(x, y, z)* on the surface of conductor *i* (*i* = *1* $\cdots$ *M*), and

$$\phi = 0, \tag{3.11}$$

for any point at infinity or at the ground plane.

The conductor charges Q_i ($i = 1 \cdots M$) are found from

$$Q_i = \int_{V_i} \rho \, dv, \tag{3.12}$$

where V_i is the volume of conductor i.

To solve 3.7 with boundary conditions 3.10 and 3.11, the boundary-element method transforms 3.7 into an integral equation, where the potential ϕ is expressed as a function of the charge density ρ over the whole domain V. This is achieved by choosing a Green's function $G(p, q)$ for V that, when no other free charge is present in V, gives the potential at a point p in V as a function of a unit point charge at a point q in V. The result is summerized as follows:

Theorem 3.1 :

Consider a domain V with an electrical permittivity ε and a boundary C. Let the radius of C approach to infinity, and let ϕ denote the potential in V and ρ the charge density in V. Let the Green's function G for V be defined such that

$$\nabla\cdot(\varepsilon \nabla G) = -\delta(p - q), \qquad p \text{ and } q \text{ in } V, \tag{3.13}$$

and

$$G = 0, \qquad p \text{ on } C. \tag{3.14}$$

(Thus, the function G is a solution for the potential distribution ϕ in V when a unit point charge is present at point q and when no other charge is present in V; compare with 3.7 and 3.11.) Then, the solution of

$$\nabla\cdot(\varepsilon \nabla\phi) = -\rho, \tag{3.15}$$

with the boundary condition

$$\phi = 0 \qquad \text{on } C, \tag{3.16}$$

is given by

$$\phi(p) = \int_V G(p, q)\, \rho(q)\, dv(q). \tag{3.17}$$

Proof:

From Green's second identity we have

$$\int_V \left(G\, \nabla\cdot(\varepsilon\nabla\phi) - \phi\, \nabla\cdot(\varepsilon\nabla G) \right) dv = \int_C \left(G\, \varepsilon \frac{\partial\phi}{\partial n} - \phi\, \varepsilon \frac{\partial G}{\partial n} \right) dl, \tag{3.18}$$

where n is the outward normal on C. Insertion of 3.13 and 3.15 into the left-hand side of 3.18 and 3.14 and 3.16 into the right-hand side of 3.18 shows

$$\phi(p) = \int_V G(p, q)\, \rho(q)\, dv(q). \tag{3.19}$$

□

For a homogeneous medium the Green's function is given by [17]

$$G(p, q) = \frac{1}{4\pi\varepsilon_1 r}, \tag{3.20}$$

where ε_1 is the permittivity of the medium, r is the distance between p and q. For a more complicated medium the Green's function can be obtained by using an images method [18] or a Fourier integral method [16]. For the medium shown in Figure 3.3, the Green's function $G(p, q)$ for both p and q in the SiO_2 layer is found as [10]

$$\begin{aligned} G(p, q) = \frac{1}{4\pi\varepsilon_1} \Bigg\{ & \frac{1}{\sqrt{z_1^2+\rho^2}} - \frac{1}{\sqrt{(2z_q+z_1)^2+\rho^2}} \\ & + \sum_{n=0}^{\infty} (-1)^n K^{(n+1)} \Bigg[\frac{1}{\sqrt{[2(n+1)d-(2z_q+z_1)]^2+\rho^2}} \\ & - \frac{1}{\sqrt{[2(n+1)d+z_1]^2+\rho^2}} + \frac{1}{\sqrt{[2(n+1)d+(2z_q+z_1)]^2+\rho^2}} \\ & - \frac{1}{\sqrt{[2(n+1)d-z_1]^2+\rho^2}} \Bigg] \Bigg\}, \end{aligned} \tag{3.21}$$

where (x_p, y_p, z_p) and (x_q, y_q, z_q) are respectively the coordinates of p and q when the origin is defined at some point on the interface between the Si layer and the SiO_2 layer, ε_1 is the permittivity of SiO_2, ε_2 is the permittivity of air or the protective coating, d is the thickness of SiO_2 layer, and

$$z_1 = z_p - z_q, \qquad \rho = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}, \qquad K = \frac{\varepsilon_1 - \varepsilon_2}{\varepsilon_1 + \varepsilon_2}. \tag{3.22}$$

The solution strategy of the boundary-element method is now as follows. Since we assume that the charge is only present on the surfaces of the conductors, 3.17 can be written in our case as

$$\phi(p) = \sum_{k=1}^{M} \int_{\Gamma_k} G(p, q)\, \sigma_k(q)\, ds_k(q), \tag{3.23}$$

where Γ_k is the surface of conductor k, and σ_k is the surface charge density on k. To approximate the real charge distribution, a set of N independent basis or shape functions $f_1 \cdots f_N$ are chosen that are defined on sub-areas $S_1 \cdots S_N$ on the conductor surfaces. The shape functions are normalized such that

$$\int_{S_i} f_i(q)\, ds_i(q) = 1 \qquad (i = 1 \cdots N). \tag{3.24}$$

Let $\{1 \cdots n_1\}$, $\{n_1 + 1 \cdots n_2\}$, ... $\{n_{M-1} + 1 \cdots N\}$ respectively denote the sets of indices of shape functions and sub-areas that are defined on conductors $1 \cdots M$. Then, the total charge distribution σ_k on conductor k is approximated by

$$\sigma_k(q) = \sum_{j=n_{k-1}+1}^{n_k} \alpha_j\, f_j(q), \tag{3.25}$$

where α_j ($j = n_{k-1}+1 \cdots n_k$) are the unknown variables to be determined. Further, insertion of 3.25 into 3.23 yields for the potential distribution ϕ,

$$\phi(p) = \sum_{k=1}^{M} \sum_{j=n_{k-1}+1}^{n_k} \alpha_j \int_{S_j} G(p, q)\, f_j(q)\, ds_j(q), \tag{3.26}$$

or equivalently,

$$\phi(p) = \sum_{j=1}^{N} \alpha_j \int_{S_j} G(p, q)\, f_j(q)\, ds_j(q). \tag{3.27}$$

Next, a set of N independent weight functions $W_1 \cdots W_N$ are introduced that are defined on each of the sub-areas $S_1 \cdots S_N$, and that are used to require

$$\int_{S_i} W_i(p) \left[\Phi_l - \sum_{j=1}^{N} \alpha_j \int_{S_j} G(p, q)\, f_j(q)\, ds_j(q) \right] ds_i(p) = 0,$$
$$(i = n_{l-1}+1 \cdots n_l,\ l = 1 \cdots M). \tag{3.28}$$

After the multiplication and reshuffling of terms, the above set of equations may be rewritten as

$$\sum_{j=1}^{N} \alpha_j \int_{S_i} \int_{S_j} G(p, q)\, f_j(q)\, W_i(p)\, ds_j(q)\, ds_i(p) = \int_{S_i} W_i(p)\, \Phi_l\, ds_i(p),$$
$$(i = n_{l-1}+1 \cdots n_l,\ l = 1 \cdots M). \tag{3.29}$$

Now, let F be an $N \times M$ incidence matrix of weight functions and conductors in which

$$F_{ij} = \begin{cases} 1 & \textit{if sub–area i is on conductor j} \\ 0 & \textit{otherwise.} \end{cases} \tag{3.30}$$

Then, Equation 3.29 may be written as a set of $N \times N$ equations,

$$G\,\alpha = W\,F\,\Phi, \tag{3.31}$$

where G is an $N \times N$ matrix that has entries

$$g_{ij} = \int_{S_i} \int_{S_j} G(p,\, q)\, f_j(q)\, W_i(p)\, ds_j(q)\, ds_i(p), \tag{3.32}$$

W is an $N \times N$ matrix that has entries

$$w_{ij} = 0, \qquad i \neq j, \tag{3.33a}$$

$$w_{ii} = \int_{S_i} W_i(p)\, ds_i(p), \tag{3.33b}$$

$\alpha^T = [\alpha_1,\, \alpha_2,\, ...,\, \alpha_N]$ and $\Phi^T = [\Phi_1,\, \Phi_2\, ...,\, \Phi_M]$.

The conductor charges $Q^T = [Q_1,\, Q_2,\, \cdots\, Q_M]$ are found from 3.31 as follows,

$$Q = F^T\,\alpha = F^T\,G^{-1}\,W\,F\,\Phi. \tag{3.34}$$

Thus, the short-circuit capacitance matrix C_s is obtained from 3.34 as

$$C_s = F^T\,G^{-1}\,W\,F. \tag{3.35}$$

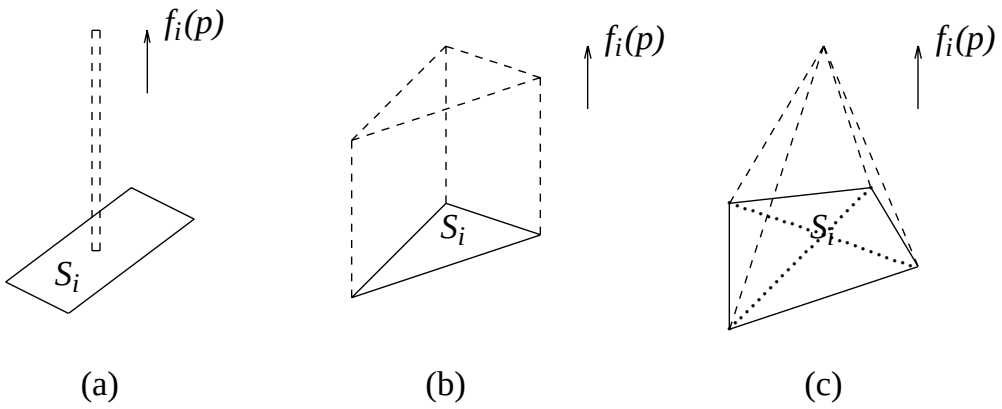
Some appropriate types of shape functions that can be chosen are shown in Figure 3.5a-c. The shape functions are ordered in increasing complexity. Figure 3.5a shows a Dirac shape function, Figure 3.5b shows a constant (zero-th order) shape function, and Figure 3.5c shows a linear (first-order) shape function. With the linear shape function a continuous charge distribution is obtained by having neighboring sub-areas overlap, such that the origin of a particular sub-area coincidences with the corners of its neighboring sub-areas.

In the Galerkin boundary-element method [10] the weight functions W_i are chosen equal to the shape functions. This way, 3.32 and 3.33 take the form

$$g_{ij} = \int_{S_i} \int_{S_j} G(p,\, q)\, f_j(q)\, f_i(p)\, ds_j(q)\, ds_i(p), \tag{3.36}$$

and

$$w_{ii} = 1. \tag{3.37}$$



(a) (b) (c)

Figure 3.5. Different types of basis or shape functions that can be used to model the surface charge density on the conductors.

Hence G is symmetrical, which reduces the number of different entries to be stored, and which provides some advantages for the solution of the capacitance matrix (see Section 3.3).

In the point-fitting boundary method [13], the weight functions W_i are chosen as Dirac functions,

$$W_i(p) = \delta(p - p_i), \tag{3.38}$$

where p_i is some position on sub-area S_i. In this case

$$g_{ij} = \int_{S_j} G(p, q)\, f_j(q)\, ds_j(q), \tag{3.39}$$

and

$$w_{ii} = 1. \tag{3.40}$$

The advantage of this method is that only single surface integrals have to be evaluated to determine G. A symmetric matrix G may be enforced in this case by replacing g_{ij} and g_{ji} by $(g_{ij} + g_{ji})/2$.

Further, the stored energy by the conductor system is given by

$$E = \tfrac{1}{2}\, \Phi^T\, C_s\, \Phi. \tag{3.41}$$

By insertion of 3.31 and 3.35 into 3.41 and by using $G = G^*$ we find that

$$E = \tfrac{1}{2}\, \alpha^T\, G\, W^{-1}\, \alpha. \tag{3.42}$$

Since for the stored energy it should hold that $E(\alpha) > 0$ for all values of $\alpha \neq 0$, we find from 3.42 that

$$\alpha^T G \alpha > 0, \tag{3.43}$$

for all values of $\alpha \neq 0$. Thus, in order to obtain a good physical model, it follows that the matrix G must be positive definite.

3.3 Approximate inversion

3.3.1 Introduction

To find the short-circuit capacitance matrix C_s, the boundary-element method computes the inverse of an elastance matrix G that has a size $N \times N$, where N is the number of different shape functions or sub-areas (see Equation 3.35). The matrix G is a full matrix and a complete inversion of it - e.g. by means of Choleski's method [19] - will require $O(N^3)$ time and $O(N^2)$ memory. However, the value of the Green's function $G(p, q)$ is inversely proportional to the distance between p and q. Therefore, many entries in G will be small compared to other entries in G. In addition, the corresponding entries in the inverse of G, which are at the same position as the small entries in G, will also be small (see the examples that are presented later in this section).

In [20, 21, 22] a matrix inversion algorithm has been described that uses the above-mentioned property of G to compute only an approximate inverse, thereby reducing the computation and space complexity of the inversion algorithm, as well as the complexity of the capacitance model. The approximate matrix inversion algorithm computes an inverse for G that has zeros at positions that correspond to the entries of G that are considered small, and non-zero values in the other positions. In order to find the approximate inverse, the method does not utilize the entries of G that are considered small.

First, in Section 3.3.2, we describe the approximate matrix inversion algorithm for the case where G is specified on a staircase band. This corresponds to computing an approximation for the solution of a one-dimensional element mesh. Next, in Section 3.3.3, we describe how the algorithm given in Section 3.3.2 is extended to compute a multiple-banded inverse. With this algorithm, an approximation for the solution of a two-dimensional or a pseudo two-dimensional element mesh is obtained (a mesh is called a pseudo two-dimensional element mesh if it is three-dimensional and if the sizes in one direction are bounded by a small constant; this for example occurs in VLSI circuits where the height of the circuit is relatively small).

3.3.2 The Schur algorithm

3.3.2.1 Description

The Schur algorithm to compute the inverse of a "maximum entropy" extension of a positive definite matrix that is specified on a staircase band was first described in [20]. Later, descriptions of it have been given in [21, 22, 23, 24]. In order to describe the algorithm, we first give some definitions.

Let G be a positive definite matrix that is specified on a staircase band S. A matrix with index pairs $\{ (i, j) \mid i = 1 \cdots N, j = 1 \cdots N \}$ is said to be specified on a staircase band S if for all $i \le k \le l \le j$ [$i \ge k \ge l \ge j$] $(i, j) \in S$ implies that $(k, l) \in S$. An example of such a matrix is given in Figure 3.7. The index pairs of the entries of G that are part of the staircase band S are, in our case, the index pairs of the entries g_{ij}, given by Equation 3.32, that are considered to be large enough to significantly contribute to the final capacitance model. Let G_{ME} denote the maximum entropy extension of the matrix G, which is defined as the matrix that is an extension of G on the unspecified part of G, such that it is positive definite, and such that it has a maximum determinant. It can be shown (see e.g. [22]) that $(G_{ME}^{-1})_{ij} = 0$ for all $(i, j) \in S$. Further, let an elementary hyperbolic rotation matrix Θ with parameters a, b and ρ and of size $2N \times 2N$ be defined as shown in Figure 3.6. The Schur algorithm to compute the triangular factors of the inverse of the maximum entropy extension of a matrix that is specified on a staircase band is then defined as follows:

Theorem 3.2 :

Let G be a positive definite matrix that is specified on a staircase band S. Let G be normalized to $G = V + I + V^*$, where V is a strictly upper triangular matrix, and let $U = V + I$. Then,

1. there are elementary hyperbolic rotation matrices $\Theta_1 \cdots \Theta_m$ such that

$$[U\ V]\ \Theta_1 \cdots \Theta_m = [U^{(m)}\ V^{(m)}], \tag{3.44}$$

   where $U^{(m)}$ is upper triangular and $V^{(m)}$ is strictly upper triangular with zeros on the strictly upper triangular part of S;

2. the product $\Theta_1 \cdots \Theta_m$ has the property that

$$[I\ I]\ \Theta_1 \cdots \Theta_m = [L_{ME}^{-*}\ M_{ME}^{-1}], \tag{3.45}$$

where L_{ME}^{-*} and M_{ME}^{-1} are the triangular factors of the inverse of a maximum entropy extension of G, i.e $G_{ME}^{-1} = L_{ME}^{-*}\ L_{ME}^{-1} = M_{ME}^{-1}\ M_{ME}^{-*}$.

For a complete proof of the above theorem we refer to [22]. Below, we will only

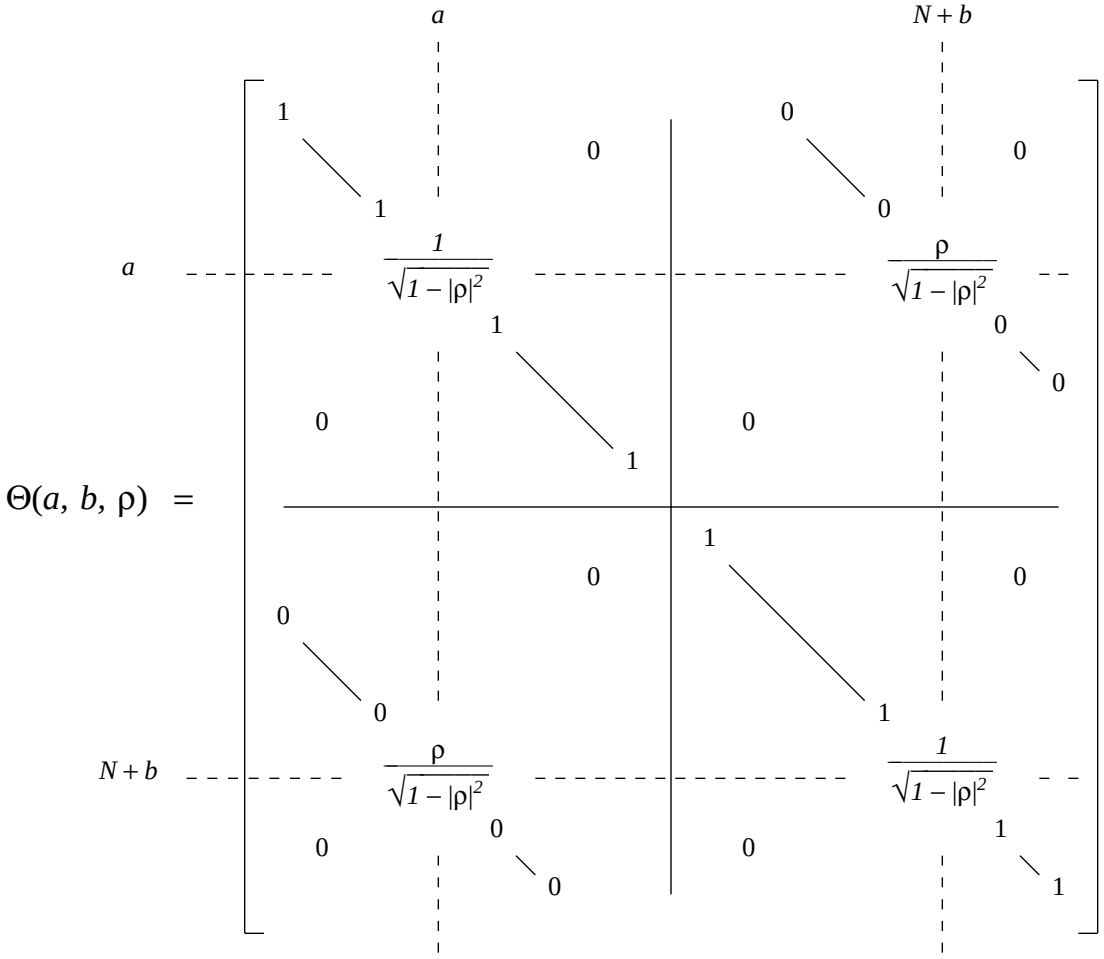


Figure 3.6. An hyperbolic rotation matrix $\Theta(a, b, \rho)$.

describe the working of the algorithm.

Let G' be an (arbitrary) positive definite matrix that is given by

$$G' = \begin{bmatrix} g'_{11} & g'_{12} & g'_{13} & \cdot & g'_{1N} \\ g'_{21} & g'_{22} & g'_{23} & \cdot & g'_{2N} \\ g'_{31} & g'_{32} & g'_{33} & \cdot & g'_{3N} \\ \cdot & \cdot & \cdot & & \cdot \\ g'_{N1} & g'_{N2} & g'_{N3} & \cdot & g'_{NN} \end{bmatrix}. \tag{3.46}$$

For the sake of simplicity, all entries g'_{ij} $(i = 1 \cdots N, j = 1 \cdots N)$ in 3.46 have been drawn, but, in fact, only some of them are specified, depending on the value of S. Since $G' = G'^{*}$ we have $g'_{ij} = \overline{g}'_{ji}$.

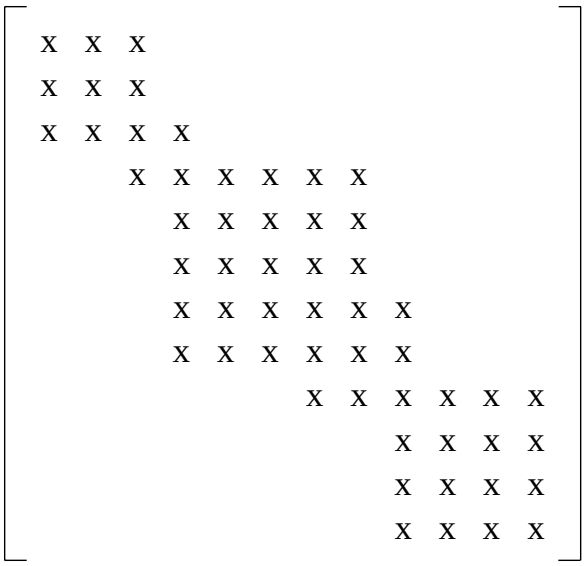


Figure 3.7. Example of a matrix that is specified on a staircase band.

First, G' is normalized to

$$G = D\,G'\,D, \tag{3.47}$$

where

$$D = \begin{bmatrix} \frac{1}{\sqrt{g'_{11}}} & 0 & 0 & . & 0 \\ 0 & \frac{1}{\sqrt{g'_{22}}} & 0 & . & 0 \\ 0 & 0 & \frac{1}{\sqrt{g'_{33}}} & . & 0 \\ . & . & . & & . \\ 0 & 0 & 0 & . & \frac{1}{\sqrt{g'_{NN}}} \end{bmatrix} \tag{3.48}$$

Hence, the entries of G are found from the entries of G' as

$$g_{ij} = \begin{cases} 1 & \text{if } i = j \\ \dfrac{g'_{ij}}{\sqrt{g'_{ii}g'_{jj}}} & \text{if } i \neq j. \end{cases} \tag{3.49}$$

As a result, the inverse of the maximum entropy extension of G' will be obtained

from the inverse of the maximum entropy extension of G as

$$G'^{-1}_{ME} = D\, G^{-1}_{ME}\, D. \tag{3.50}$$

Next, G is decomposed to $G = V + I + V^*$, where V is a strictly upper triangular matrix, and $U = V + I$. Then, $[\, U\, V\,]$ is written as

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} & . & u_{1N} & 0 & v_{12} & v_{13} & . & v_{1N} \\ 0 & u_{22} & u_{23} & . & u_{2N} & 0 & 0 & v_{23} & . & v_{2N} \\ 0 & 0 & u_{33} & . & u_{3N} & 0 & 0 & 0 & . & v_{3N} \\ . & . & . & & . & . & . & . & & . \\ 0 & 0 & 0 & . & u_{NN} & 0 & 0 & 0 & . & 0 \end{bmatrix}, \tag{3.51}$$

where $u_{ij} = v_{ij} = g_{ij}$.

Let $[\, U^{(n)}\, V^{(n)}\,]$ denote the result of a multiplication of $[\, U\, V\,]$ with a set of elementary hyperbolic rotation matrices $\Theta_1 \cdots \Theta_n$, each having a form as given by 3.6. Then, the entries of $[\, U^{(n)}\, V^{(n)}\,]$ are obtained from the entries of $[\, U^{(n-1)}\, V^{(n-1)}\,]$, after multiplication with a rotation matrix $\Theta_n(a, b, \rho_{ab})$, as

$$u_{ij}^{(n)} = \frac{u_{ij}^{(n-1)} + \rho_{ab} v_{ib}^{(n-1)}}{\sqrt{1 - |\rho_{ab}|^2}} \qquad \text{for } j = a, \tag{3.52a}$$

$$u_{ij}^{(n)} = u_{ij}^{(n-1)} \qquad \text{for } j \neq a, \tag{3.52b}$$

$$v_{ij}^{(n)} = \frac{v_{ij}^{(n-1)} + \rho_{ab} u_{ia}^{(n-1)}}{\sqrt{1 - |\rho_{ab}|^2}} \qquad \text{for } j = b, \tag{3.52c}$$

$$v_{ij}^{(n)} = v_{ij}^{(n-1)} \qquad \text{for } j \neq b. \tag{3.52d}$$

The Schur algorithm now proceeds by choosing the set of hyperbolic rotation matrices $\Theta_1 \cdots \Theta_m$ such that each Θ_n $(n = 1 \cdots m)$ subsequently eliminates a non-zero entry of V, until a matrix $[\, U^{(m)}\, V^{(m)}\,]$ is obtained in which $V^{(m)}$ has zeros on the strictly upper triangular part of S. It is possible to do so when first eliminating all relevant entries at the first upper diagonal, then eliminating all relevant entries at the second upper diagonal, etc.

As an example we consider the elimination of v_{12} and v_{23}.

Suppose $(1, 2) \in S$. Then, when using $a = 1$ and $b = 2$, the reflection coefficient ρ_{12} for Θ_1 to eliminate v_{12} is chosen as

$$\rho_{12} = -\frac{v_{12}}{u_{11}}. \tag{3.53}$$

In this case, it follows from 3.52 that $v_{12}^{(1)} = 0$, while U remains upper triangular and V remains strictly upper triangular.

Second, when $(2, 3) \in S$, we eliminate v_{23} with Θ_2 by using $a = 1$, $b = 2$ and

$$\rho_{23} = -\frac{v_{23}^{(1)}}{u_{22}^{(1)}}. \tag{3.54}$$

Again, U remains upper triangular and V remains strictly upper triangular.

The process is continued with ρ_{34} to eliminate v_{34}, ρ_{45} to eliminate v_{45}, etc., and ρ_{13} to eliminate v_{13}, etc., until all relevant entries that correspond to positions on the strictly upper triangular part of S have been eliminated.

In general, entry $v_{ij}^{(n)}$ of $V^{(n)}$ is eliminated with Θ_n, which has $a = i$, $b = j$ and

$$\rho_{ij} = -\frac{v_{ij}^{(n)}}{u_{ii}^{(n)}}. \tag{3.55}$$

The matrices Θ_n $(n = 1 \cdots m)$ with parameter sets $\{a_n, b_n, \rho_{a_n b_n}\}$ $(n = 1 \cdots m)$ to eliminate the entries v_{ij} $((i, j) \in S)$ of V, are ordered such that the values $b_n - a_n$ $(n = 1 \cdots N)$ occur in non-decreasing order.

It can be shown [22] that the product $\Theta_1 \cdots \Theta_m$ can be written as

$$\begin{bmatrix} U_{ME}^{*} & -V_{ME}^{*} \\ -V_{ME}^{*} & U_{ME} \end{bmatrix} \begin{bmatrix} L_{ME}^{-*} & 0 \\ 0 & M_{ME}^{-1} \end{bmatrix}, \tag{3.56}$$

where L_{ME}^{-*} and M_{ME}^{-1} are the triangular factors of G_{ME}. Hence it follows that

$$[I\ I]\ \Theta_1 \cdots \Theta_m = [L_{ME}^{-*}\ M_{ME}^{-1}]. \tag{3.57}$$

3.3.2.2 Physical interpretation

The approximate inverse as provided by the Schur algorithm is the inverse of a maximum entropy extension of G, which is the unique positive definite matrix that coincides with G on S and of which the inverse is zero on the complement of S. This results in the following property for the physical model that is provided by

the Schur algorithm (first we give two definitions).

Definition 3.1 :

Consider a boundary-element capacitance model as given in Section 3.2.2 with sub-areas $1 \cdots N$, an elastance matrix G, and a weight matrix W (see 3.31). The network with nodes $1 \cdots N$, node charges $\alpha_1 \cdots \alpha_N$ and node potentials $\phi_1 \cdots \phi_N$, that accords to the set of simultaneous equations

$$\alpha = G^{-1} W \phi, \tag{3.58}$$

where $\alpha = [\alpha_1, \alpha_2, \cdots, \alpha_N]^T$ and $\phi = [\phi_1, \phi_2, \cdots, \phi_N]^T$, is called the elementary capacitance network.

Definition 3.2 :

Consider a boundary-element capacitance model as given in Section 3.2.2 with sub-areas $1 \cdots N$, an elastance matrix G, and a weight matrix W. Let G be specified on a staircase band S, and let the maximum entropy extension of this matrix be given by G_{ME}. The network with nodes $1 \cdots N$, node charges $\alpha_1 \cdots \alpha_N$ and node potentials $\phi_1 \cdots \phi_N$, that obeys to the set of simultaneous equations

$$\alpha = G_{ME}^{-1} W \phi, \tag{3.59}$$

where $\alpha = [\alpha_1, \alpha_2, \cdots, \alpha_N]^T$ and $\phi = [\phi_1, \phi_2, \cdots, \phi_N]^T$, is called an approximation of the elementary capacitance network for the staircase band S.

Theorem 3.3 :

Consider an elementary capacitance network with nodes $1 \cdots N$ and consider an approximation of the elementary capacitance network for a staircase band S. Then, if $(a, b) \in S$, with $a < b$, and if in both networks nodes $\{ i \mid 1 \le i < a \text{ or } b < i \le N\}$ are kept chargeless, the relation between the node potentials $\phi_a \cdots \phi_b$ and the node charges $\alpha_a \cdots \alpha_b$ in the elementary capacitance network is identical to the relation between the node potentials $\phi_a \cdots \phi_b$ and the node charges $\alpha_a \cdots \alpha_b$ in the approximation of the elementary capacitance network.

Proof:

Let the entries of G be given by $\{g_{ij} \mid i = 1 \cdots N, j = 1 \cdots N\}$. Let the entries of a maximum entropy extension G_{ME} for G when G is specified on a staircase band S, be given by $\{\tilde{g}_{ij} \mid i = 1 \cdots N, j = 1 \cdots N\}$. Clearly,

$$g_{ij} = \tilde{g}_{ij} \qquad (i = 1 \cdots N, j = 1 \cdots N \wedge (i, j) \in S). \tag{3.60}$$

The relation between the node potentials $\phi_1 \cdots \phi_N$ and the node charges $\alpha_1 \cdots \alpha_N$ for the elementary capacitance network is given by

$$\phi_i = \frac{1}{w_{ii}} \sum_{j=1}^{N} g_{ij}\, \alpha_j \qquad (i=1 \cdots N). \tag{3.61}$$

The relation between the node potentials $\tilde{\phi}_1 \cdots \tilde{\phi}_N$ and the node charges $\tilde{\alpha}_1 \cdots \tilde{\alpha}_N$ for the approximation of the elementary capacitance network is given by

$$\tilde{\phi}_i = \frac{1}{w_{ii}} \sum_{j=1}^{N} \tilde{g}_{ij}\, \alpha_j \qquad (i=1 \cdots N). \tag{3.62}$$

Now, consider a pair of nodes (a, b) for which $(a, b) \in S$. The nodes $\{ j \mid 1 \leq j < a \text{ or } b < j \leq N \}$ in both networks are free of charge, that is

$$\alpha_j = 0 \qquad (j = 1 \cdots N, j < a \text{ or } j > b), \tag{3.63}$$

and

$$\tilde{\alpha}_j = 0 \qquad (j = 1 \cdots N, j < a \text{ or } j > b). \tag{3.64}$$

For $\phi_a \cdots \phi_b$ we find by the insertion of 3.63 into 3.61,

$$\phi_i = \frac{1}{w_{ii}} \sum_{j=a}^{b} g_{ij}\, \alpha_j \qquad (i = a \cdots b). \tag{3.65}$$

For $\tilde{\phi}_a \cdots \tilde{\phi}_b$ we find by the insertion of 3.64 into 3.62, and using 3.60,

$$\tilde{\phi}_i = \frac{1}{w_{ii}} \sum_{j=a}^{b} g_{ij}\, \tilde{\alpha}_j \qquad (i = a \cdots b). \tag{3.66}$$

Thus, it follows that the relation between the potentials and the charges of nodes $a \cdots b$ in the elementary capacitance network is identical to the relation between the potentials and the charges of the nodes $a \cdots b$ in the approximation of the elementary capacitance network.

□

The above is illustrated by the example shown in Figure 3.8, which consists of a (theoretical) situation where there are four sub-areas/nodes above a ground plane. Suppose, the elastance matrix G for the model of Figure 3.8 is given by

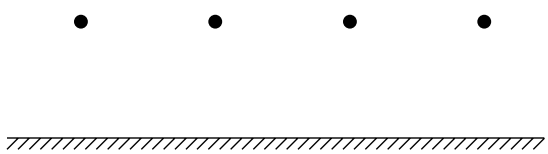


Figure 3.8. Four sub-areas/nodes above a ground plane.

$$G = \begin{bmatrix} 1.0 & 0.4 & 0.2 & 0.1 \\ 0.4 & 1.0 & 0.4 & 0.2 \\ 0.2 & 0.4 & 1.0 & 0.4 \\ 0.1 & 0.2 & 0.4 & 1.0 \end{bmatrix}, \tag{3.67}$$

and suppose that the incidence matrix F and the weight matrix W both equal the identity matrix. A full inversion of G gives

$$C_s = G^{-1} = \begin{bmatrix} 1.193 & -0.454 & -0.054 & -0.007 \\ -0.454 & 1.366 & -0.434 & -0.054 \\ -0.054 & -0.434 & 1.366 & -0.454 \\ -0.007 & -0.054 & -0.454 & 1.193 \end{bmatrix}. \tag{3.68}$$

The network that corresponds to the short-circuit capacitance matrix of 3.68 is shown in Figure 3.9a. In Figure 3.9b-d approximations to the network of Figure 3.9a that have been obtained by using the Schur algorithm to compute an approximate inverse with a support as indicated are shown. Because of Theorem 3.3, the capacitance to ground for each node, with the other nodes free of charge, is equal in Figure 3.9a-d. Also because of Theorem 3.3, the relation between the potentials and charges of neighboring nodes, with the other nodes chargeless, is equal in Figure 3.9a-c, and the relation between the potentials and charges of triplets of adjacent nodes, with the other nodes chargeless, is equal in Figure 3.9a-b.

A theorem similar to Theorem 3.3, but now for the conductor capacitances, cannot be given. This is because nodes of one mesh that are located on the same conductor are galvanically connected, and hence a charge at one particular node will induce a charge displacement at all other nodes. Thus, the charge of none of the nodes can be guaranteed to be equal to zero, and a similar reasoning as that given in the proof of Theorem 3.3 is not found. However, it is expected that Theorem 3.3 gives a reasonable approximation for the interconnect capacitances when replacing ‘sub-areas’ and ’nodes’ by ‘conductors’ (see also the example that is presented in the following section).

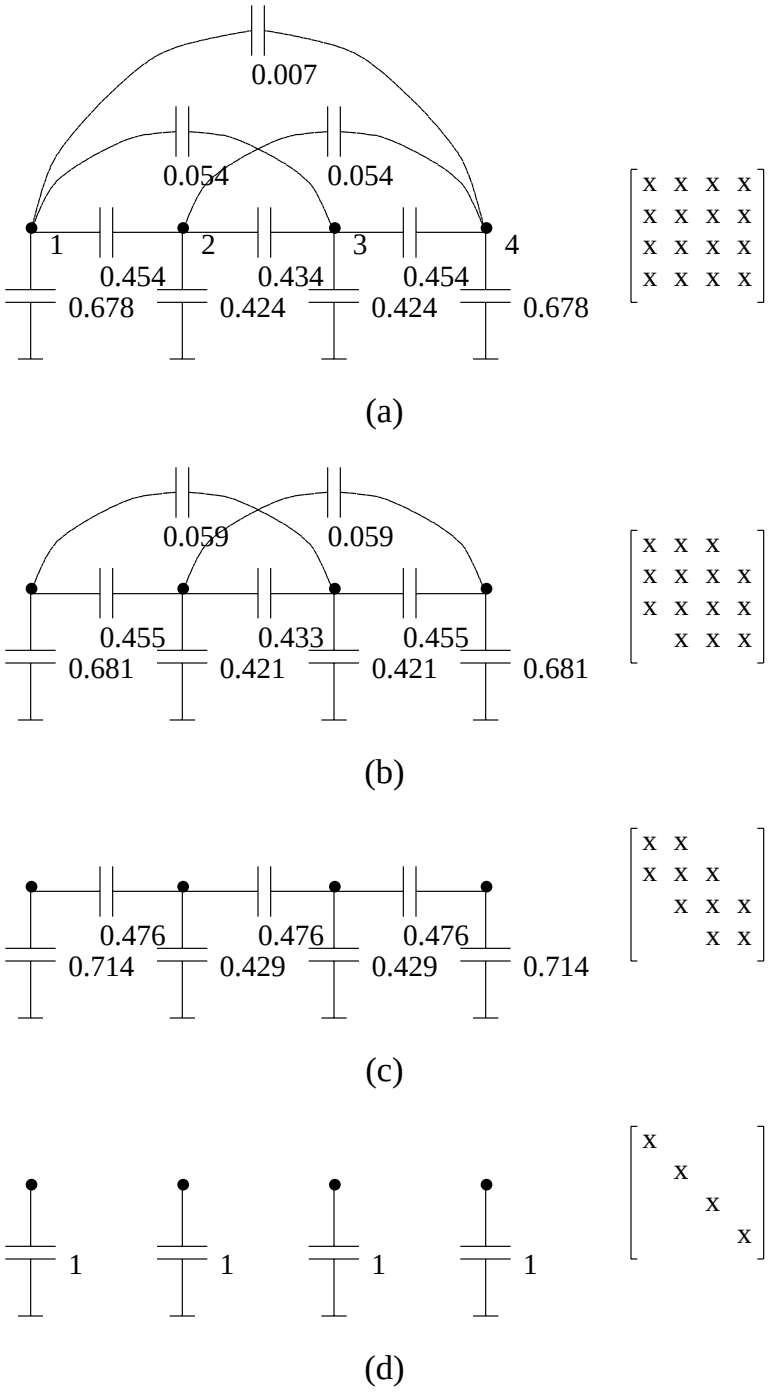


Figure 3.9. (a) Exact solution for the model as shown in Figure 3.8, (b) approximate solution in which diagonals 1-3 have been computed, (c) approximate solution in which diagonals 1-2 have been computed, (d) approximate solution in which only the main diagonal has been computed.

3.3.2.3 Example

We illustrate the application of the Schur algorithm by computing the capacitances for a conductor configuration that consists of 5 parallel conductors as shown by Figure 3.10. The capacitance model used is a point fitting boundary-element model in which the conductor charges are represented by line charges that are present on edges between nodes on the conductor surfaces [13]. For the configuration shown in Figure 3.10, the mesh has 220 nodes, located 1 μ apart on the top and bottom of the conductors.

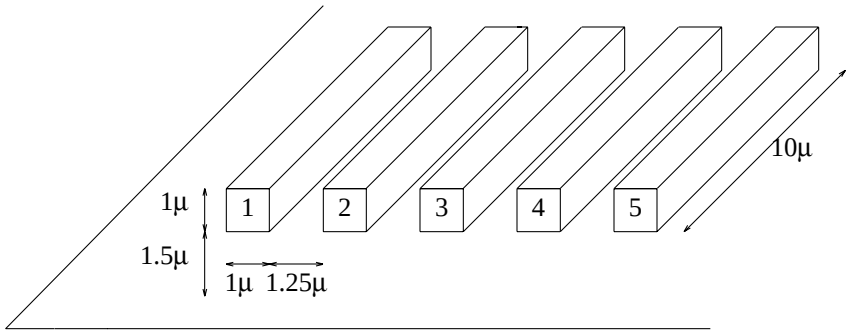


Figure 3.10. 5 parallel conductors above a ground plane.

For the first experiment the nodes are numbered as indicated in Figure 3.11. The support S of the elastance matrix G is chosen such that only influences are taken into account between nodes that are in the x direction within a distance w of each other. The results are presented in Table 3.1. For different window sizes, the different network capacitances for the first conductor are shown as well as the total capacitance to ground for the first conductor when the other conductors are kept free of charge (C_{f11}), and the total capacitance to ground for the first conductor when the other conductors are short-circuited to ground (the short-circuit capacitance C_{s11}).

Note that when decreasing the size over which influences are taken into account, the loss of the number of coupling capacitances is compensated for by an increase in the value of the other network capacitances, such that C_{f11} is approximately constant. Note that also the short-circuit capacitance C_{s11} is approximately constant.

The second experiment uses a numbering scheme and a window as indicated in Figure 3.12. In this example, the support S of the elastance matrix G is chosen such that only influences are taken into account between nodes that within a distance w in the y direction. Thus, the number of different network coupling capacitances that are computed in this case is independent of the size of the

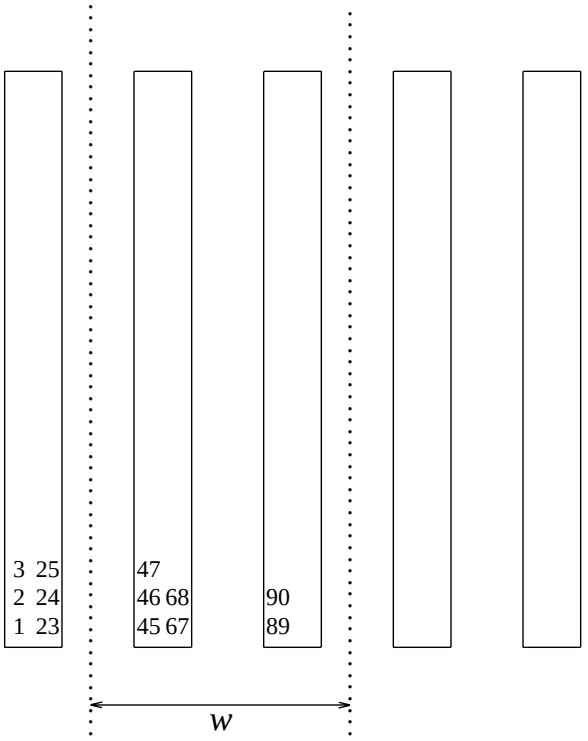


Figure 3.11. Numbering scheme and window for the first experiment.

Table 3.1. Capacitance values when using the numbering scheme of Figure 3.11.

	capacitances (10^{-18} F)						
w (μ)	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{f11}	C_{s11}
10	1128	557.0	37.59	14.32	9.13	1562	1745
7.75	1134	557.1	37.83	17.34	0	1562	1745
5.5	1144	557.6	43.65	0	0	1562	1745
3.25	1172	572.7	0	0	0	1561	1745
1	1519	0	0	0	0	1519	1519

window. However, when decreasing the size of the window, the accuracy decreases because less coupling effects are computed between nodes that are far from each other in the y direction. The net effect is an increase in the value of the ground capacitances and a decrease in the value of the coupling capacitances. Again, note that the capacitances C_{f11} and C_{s11} are approximately constant.

As a conclusion of both experiments, we may find that a reasonable degree of accuracy is obtained when using the Schur algorithm for window sizes that are in the same order of magnitude as the vertical dimensions of the circuit, and that the capacitances C_f and C_s of a conductor - which largely determine the first-order

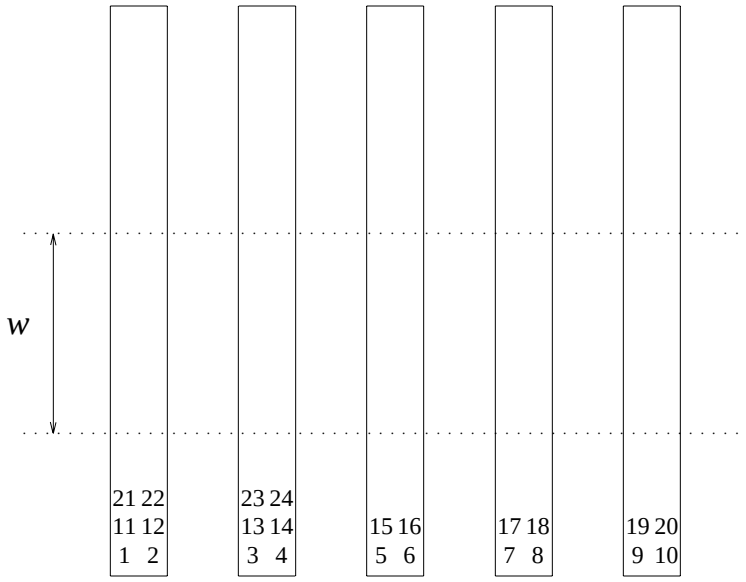


Figure 3.12. Numbering scheme and window for the second experiment.

Table 3.2. Capacitance values when using the numbering scheme of Figure 3.12.

	capacitances (10^{-18} F)						
w (μ)	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{f11}	C_{s11}
10	1128	557.0	37.59	14.93	9.13	1562	1746
8	1129	556.8	37.46	14.23	9.04	1563	1746
6	1132	556.1	37.00	13.93	8.75	1565	1748
4	1142	554.1	35.85	13.27	8.19	1573	1754
3	1154	551.7	34.86	12.78	7.81	1583	1762
2	1180	547.4	33.68	12.25	7.37	1616	1781

timing behavior of the conductor - are only slightly dependent on the size of the window.

3.3.3 An extension of the Schur algorithm

In the previous section it was shown how the Schur algorithm can be used to find the approximate inverse for an elastance matrix G that is specified on a staircase band S. For an arbitrary element mesh of a dimension > 1, however, it is not possible to choose the staircase band S such that the indices of all entries of G that correspond to node pairs of which the nodes are within a distance d are in the staircase band, and the indices of all entries of G that correspond to node pairs of which the nodes are further apart are not in the staircase band. Therefore, in [21] an extension of the Schur algorithm has been described that can be used to find the approximate inverse for a matrix that is specified on a multiple band. For a two-

dimensional or a pseudo two-dimensional element mesh this algorithm finds an approximate inverse in which all influences between nodes that are within a distance *d* are taken into account, and all influences between nodes that are further apart are not. Thus, an efficient solution of the boundary-element model is obtained for each conductor configuration, no matter its horizontal sizes.

3.3.3.1 Description

The extension of the Schur algorithm to compute the inverse of a matrix that is specified on a multiple band has been described in [21, 22]. The approximate inverse that is provided is the inverse of a matrix that closely matches the partially specified matrix. It has zeros at all positions that correspond to unspecified entries in the partially specified matrix. In order to give the definition of the algorithm, we first introduce some notations.

Let $\mathbf{G}$ be a matrix that is partitioned as a block matrix $\mathbf{G} = [\ \mathbf{g}_{ij}\ ]$, $i = 1 \cdots L$, $j = 1 \cdots L$, where $\mathbf{g}_{ij}$ denotes a block of $\mathbf{G}$ (see Figure 3.13). Let $\mathbf{G}(i, j)$ denote the principle block matrix that lies in the rows and columns of $\mathbf{G}$ indexed by $i \cdots j$ (see also Figure 3.13). The matrix $\square[\mathbf{G}; (i, j)]$ is defined as the matrix that satisfies $(\square[\mathbf{G}; (i, j)])(i, j) = \mathbf{G}$ and that is zero on the other parts of $\square[\mathbf{G}; (i, j)]$. The size of $\square[\mathbf{G}; (i, j)]$ is determined by its context. In other words, $\mathbf{G}(., .)$ takes a block matrix out of a matrix $\mathbf{G}$, and $\square[\mathbf{G}; (., .)]$ embeds a matrix $\mathbf{G}$ in a zero matrix.

Further, we remark that if a positive definite block matrix $\mathbf{G}$ can be permuted to a positive definite block matrix $P\ \mathbf{G}\ P^*$ - where P is a permutation matrix - that is specified on a staircase band, then the inverse of the maximum entropy extension of $\mathbf{G}$ is obtained from

$$\mathbf{G}_{ME}^{-1} = P^*\ (\ P\ \mathbf{G}\ P^*\ )_{ME}^{-1}\ P. \tag{3.69}$$

The extension of the Schur algorithm to compute the inverse of a matrix that is specified on a multiple band is then defined as follows [21].

Definition 3.3 :

Let $\mathbf{G}$ be a positive definite matrix that is specified on a multiple band, and let it be partitioned as $\mathbf{G} = [\ \mathbf{g}_{ij}\ ]$, $i = 1 \cdots L$, $j = 1 \cdots L$. The blocks $\mathbf{g}_{ij}$, $i = 1 \cdots L$, $j = 1 \cdots L$ are such that for some block band B with support $\{(i, j)\ |\ |\ i - j\ | \leq b\ \}$ (1) the partially specified principle submatrices $\mathbf{G}(j, j + b)$, $j = 1 \cdots L–b$, can be permuted to positive definite matrices that are specified on a staircase band, and (2) the blocks on the complement of B are unspecified. Then, the sparse-inverse approximation of $\mathbf{G}$, denoted by $\mathbf{G}_{SI}$, is defined such

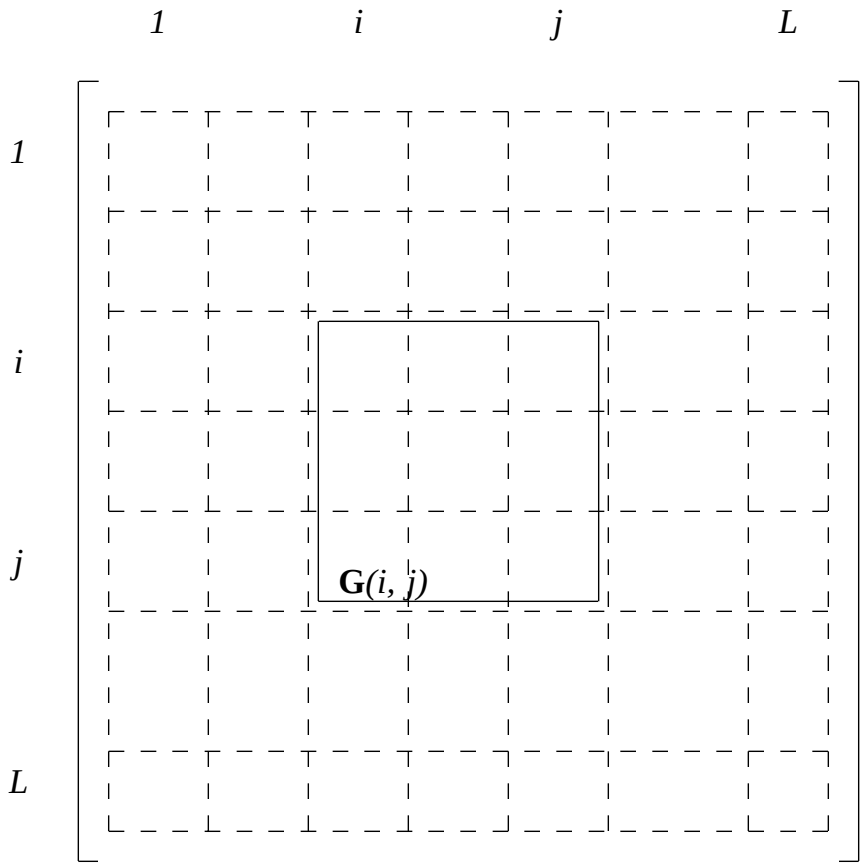


Figure 3.13. A block matrix **G** and one of its principle block matrices **G**(*i*, *j*).

that

$$\mathbf{G}_{SI}^{-1} = \sum_{j=1}^{L-b} \square[\mathbf{G}(j, j+b)_{ME}^{-1}; (j, j+b)] \tag{3.70a}$$

$$- \sum_{j=1}^{L-b-1} \square[\mathbf{G}(j+1, j+b)_{ME}^{-1}; (j+1, j+b)]. \tag{3.70b}$$

For a more extensive discussion on the properties of $\mathbf{G}_{SI}$ and $\mathbf{G}_{SI}^{-1}$, e.g. a comparison between **G** and $\mathbf{G}_{SI}$, and $\mathbf{G}^{-1}$ and $\mathbf{G}_{SI}^{-1}$, see [22].

As an example of the determination of the inverse of a matrix that is specified on a multiple band, we consider the block matrix given in Figure 3.14, which has $L = 3$ and $b = 1$. From Definition 3.3 it follows that the inverse of the sparse-inverse approximation of this matrix is given by

$$\mathbf{G}_{SI}^{-1} = \square[\mathbf{G}(1,2)_{ME}^{-1};(1,2)] + \square[\mathbf{G}(2,3)_{ME}^{-1};(2,3)] - \square[\mathbf{G}(2,2)_{ME}^{-1};(2,2)], \quad (3.71)$$

where $\mathbf{G}(1,2)_{ME}^{-1}$ and $\mathbf{G}(2,3)_{ME}^{-1}$ are obtained by permuting $\mathbf{G}(1,2)$ and $\mathbf{G}(2,3)$ respectively to matrices that are specified on a staircase band (see Figure 3.15).

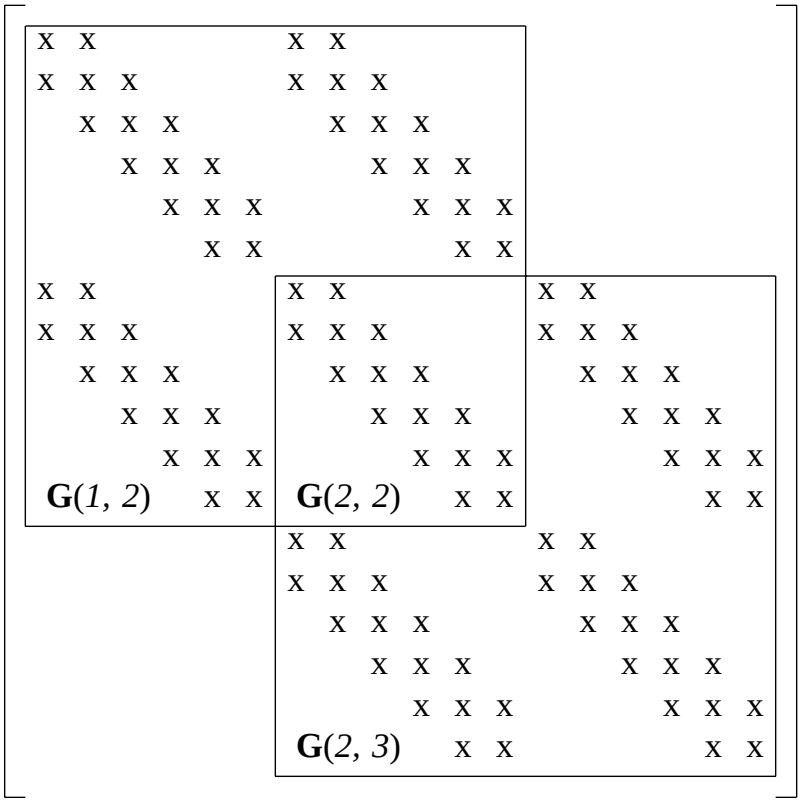


Figure 3.14. Example of a matix that is specified on a multiple band.

3.3.3.2 Example

In the following, we will show how the extension of the Schur algorithm is used to efficiently solve the boundary-element model of the conductor configuration given in Figure 3.10. In order to obtain the desired approximation, the layout of the conductor configuration is first subdivided into a set of vertical strips that have a width w_x (see Figure 3.16). Then, an approximate inverse is computed with the Schur algorithm as described in Section 3.3.2 for each strip that is not the first strip or the last strip. Hereby a numbering scheme and a window of width w_y in the y direction are used as shown in Figure 3.16. The approximate inverses that are obtained in this way correspond to the separate terms in the summation of 3.70b. Next, an approximate inverse is computed for each pair of neighboring

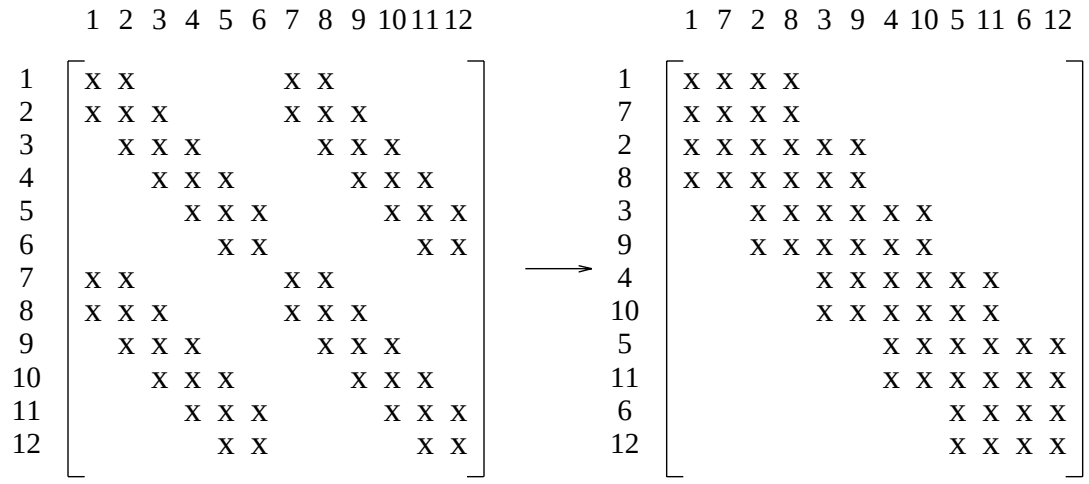


Figure 3.15. Permuting a matrix that is specified on a multiple band to a matrix that is specified on a staircase band (rows/columns *1, 2, 3, 4, 5,* $\cdots$ *12* are reordered as *1, 7, 2, 8, 3* $\cdots$ *12*).

strips, thereby using a similar window in the y direction as for the single strips and a numbering scheme as depicted in Figure 3.17. The approximate inverses that are found in this way correspond to the separate terms in the summation of 3.70a. Finally, all approximate inverses are added according to 3.70 and the result is used to compute the capacitances of the conductors.

In Table 3.3 the results are presented that were obtained using the above-mentioned technique. From Table 3.3 we see that the errors that are found when using the extension of the Schur algorithm are in the same order of magnitude as those found when using the exact Schur algorithm. Note that (again) the virtual ground capacitances C_{f11} and C_{s11} are only slightly dependent on the size of the window.

3.4 Solution scheme

In this section we will discuss the solution scheme of the boundary-element method for capacitance extraction. First, in Section 3.4.1, the solution scheme of the Schur algorithm is studied. It shown how parallelism of the algorithm can be used to speed up the computation times for the algorithm, and how an incremental solution scheme assures a low memory usage for the algorithm. Second, in Section 3.4.2, it is explained how the boundary-element model and the Schur algorithm are used to efficiently compute the interconnect capacitances of an arbitrary VLSI circuit. In this section also some results are presented for the extraction method with respect to computation times and memory usage.

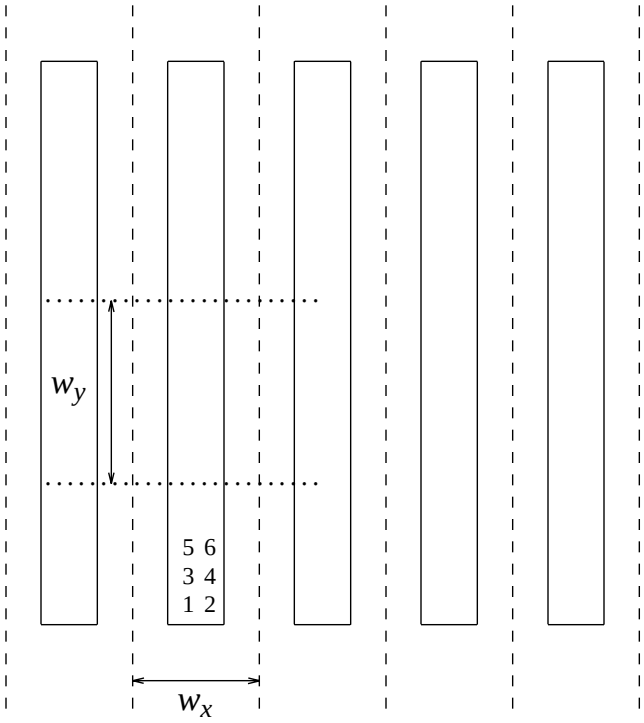


Figure 3.16. Subdividing the layout shown in Figure 3.10 into vertical strips, and the numbering scheme and the window within the second strip to compute $\mathbf{G}(2, 2)^{-1}_{ME}$.

3.4.1 Solution scheme of the Schur algorithm

To study the complexity of the Schur algorithm we consider its dependence graph(s). A dependence graph of an algorithm is a graph consisting of nodes and directed edges, where the nodes represent basic (or elementary) operations of the algorithm and the edges represent the transportation of the variables of the algorithm between these operations. From a dependence graph it is possible to determine which operations can be executed in parallel, and how the algorithm may be implemented on a processor array [25, 26, 27], or in a computer language (e.g. C or Pascal).

Suppose the upper triangular part of an elastance matrix G is given by (only the entries of which the indices are part of the staircase band S are drawn)

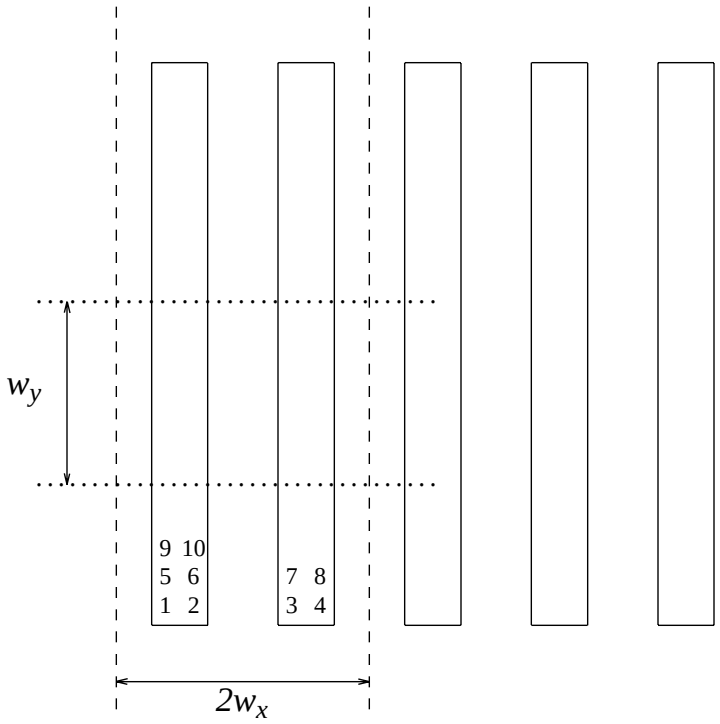


Figure 3.17. Numbering scheme and window to compute $\mathbf{G}(1, 2)^{-1}_{ME}$.

Table 3.3. Capacitance values when using the numbering scheme of Figure 3.16 and 3.17.

	capacitances (10^{-18} F)						
w_x, w_y (μ)	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{f11}	C_{s11}
10	1128	557.0	37.59	14.93	9.13	1562	1746
4.4	1148	554.1	36.06	15.87	0	1573	1751
2.2	1220	560.1	0	0	0	1606	1780

$$\begin{bmatrix} g_{11} & g_{12} & g_{13} & g_{14} & \\ & g_{22} & g_{23} & g_{24} & g_{25} \\ & & g_{33} & g_{34} & g_{35} \\ & & & g_{44} & g_{45} \\ & & & & g_{55} \end{bmatrix}. \tag{3.72}$$

A dependence graph to determine all reflection coefficients that are used to eliminate the entries v_{pq} for which $(p, q) \in S$ is shown in Figure 3.18. In this dependence graph, which has coordinate axes i, j and k, the entries u_{pq} and v_{pq} for

which $(p, q) \in S$, are input at the bottom plane ($i = 1$). The reflection coefficients ρ_{12}, ρ_{23}, ρ_{34} and ρ_{45} to eliminate the entries v_{12}, v_{23}, v_{34} and v_{45}, are computed at the grid points $\{(i, j, k) \mid i = 1, j = 1, k = 1 \cdots 4\}$. These reflection coefficients are transported to the (other) grid points in the plane $i = 1$ to obtain a matrix $[\, U^{(n)}\ V^{(n)}\,]$ where $V^{(n)}$ has its first upper diagonal filled with zeros. Next, in the middle grid plane, the reflection coefficients ρ_{13}, ρ_{24}, ρ_{35} are computed to obtain a matrix $[\, U^{(n)}\ V^{(n)}\,]$ where $V^{(n)}$ has zero entries in the first and second upper diagonal. Finally, in the top plane, the reflection coefficients ρ_{14}, ρ_{25} are computed to obtain a matrix $[\, U^{(n)}\ V^{(n)}\,]$ where $V^{(n)}$ has zeros also on the third upper diagonal.

In general, at each grid point $(i, 1, k)$ in Figure 3.18 that is hit by a variable u_{in} and a variable v_{in}, a reflection coefficient ρ is determined from

$$\rho := -\frac{v_{in}}{u_{in}}, \tag{3.73}$$

and at each grid point (i, j, k) that is hit by a variable u_{in} and a variable v_{in}, a hyperbolic rotation is computed according to

$$u_{out} := \frac{u_{in} + \rho\, v_{in}}{\sqrt{1 - |\rho|^2}}, \qquad v_{out} := \frac{v_{in} + \rho\, u_{in}}{\sqrt{1 - |\rho|^2}}, \tag{3.74}$$

where the reflection coefficient ρ at grid point (i, j, k) is obtained from the reflection coefficient that is computed at grid point $(i, 1, k)$.

A dependence graph to determine the triangular factors of the maximum entropy extension of G according to the reflection coefficients that have been found in Figure 3.18 is shown in Figure 3.19. In this figure, the reflection coefficients that have been found in Figure 3.18 are input at the front of the dependence graph ($j = 4$). The entries of $[\, U\ V\,]$ that are input in the dependence graph shown in Figure 3.18 are replaced in the dependence graph shown in Figure 3.21 by the entries of the matrix $[\, I\ I\,]$, where I is an identity matrix that has the same size as U or V. The outputs of this dependence graph are the matrices L_{ME}^{-*} and M_{ME}^{-1}, where L_{ME}^{-*} is lower triangular and entry l_{pq} of L_{ME}^{-*} is output at grid position $(i, p - q + 1, q)$, and M_{ME}^{-1} is upper triangular and entry m_{pq} of M_{ME}^{-1} is output at grid position $(i, p - q + 1 + i, q - i)$.

3.4.1.1 Mapping on processor arrays

One way to implement the Schur algorithm based on the dependence graphs shown in Figure 3.18 and Figure 3.19 is by using a hardware implementation, where the Schur algorithm is executed on an array of identical processors. A

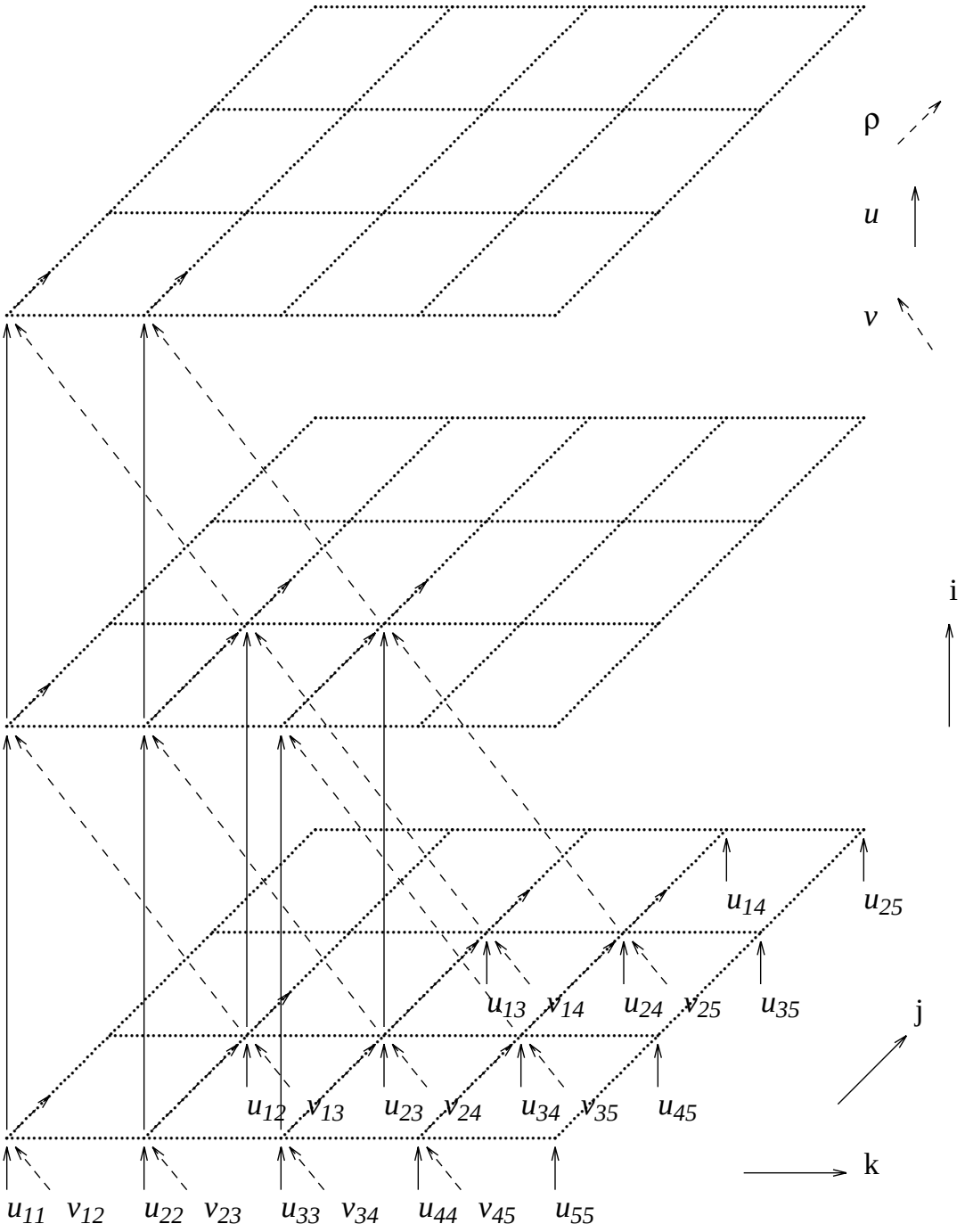


Figure 3.18. Dependence graph for computing the reflection coefficients.

processor that may be used to construct such an array is shown in Figure 3.20. It has inputs u_{in}, v_{in} and ρ_{in}, outputs u_{out}, v_{out} and ρ_{out}, and it uses a control variable c. The behavior of the processor is defined as indicated in Figure 3.20. In order to use the processor for the execution of the dependence graphs shown in Figure 3.18 and Figure 3.19, the control variable c is set to *1* for grid points in Figure 3.18 for which $j = 1$, and it is set to *0* for the remaining grid points. The mapping of iterative algorithms onto processor arrays is described in more detail in e.g. [26, 27]. It consists of first choosing sets of operations that can be executed in parallel (e.g. in Figure 3.18 the operations that are in one plane that is perpendicular to the normal vector $[1, 2, -2]^T$ can be executed in parallel), and then, by means of partitioning and clustering, allocating the different operations to the different time slots that are available for the different processors of the processor array. A processor array that might for example be used for this is the CORDIC processor as described in [28], which is capable - by means of parallelization and pipelining - of performing 10^7 basic operations per second.

3.4.1.2 Computer algorithm

When implementing the Schur algorithm in a sequential computer algorithm, the i, j and k variables act as the for-loop variables that enumerate all operations of the algorithm. An example of such an implementation is given by Algorithm 3.1. In this algorithm, the upper triangular part of a normalized positive definite matrix G is stored in a two-dimensional array $g[][]$, such that g_{pq} corresponds to $g[p][q - p + 1]$. The array $w[]$ specifies the support of G and it is defined such that g_{pq} is unspecified for $q > p + w[p]$ or $p > q + w[q]$, and g_{pq} is specified in all other cases. The entries that are computed for L_{ME}^{-*} and M_{ME}^{-1} are stored in arrays that are denoted by $\hat{l}[]$ and $\hat{m}[]$. The entry $\hat{l}_{pq}$ of L_{ME}^{-*} corresponds to $\hat{l}[q][p - q + 1]$, and the entry $\hat{m}_{pq}$ of M_{ME}^{-1} corresponds to $\hat{m}[q][q - p + 1]$. The intermediate results of U, V, L_{ME}^{-*} and M_{ME}^{-1} are maintained in two-dimensional arrays that are respectively called u, v, l and m. During each iteration of i in Algorithm 3.1, one upper diagonal of $V^{(n)}$ is eliminated and the intermediate result for the matrices L_{ME}^{-*} and M_{ME}^{-1} is computed.

3.4.1.3 Vectorization

The execution time of Algorithm 3.1 on a computer may be accelerated by running the program on a computer that is capable of performing vector operations [29]. The compiler of such a computer will search the program for operations within for-loops that can be vectorized. In our case these operations are the array operations in the two inner loops of the algorithm. For the array operations in the two inner loops, the left-hand side variables are stored in positions that are not

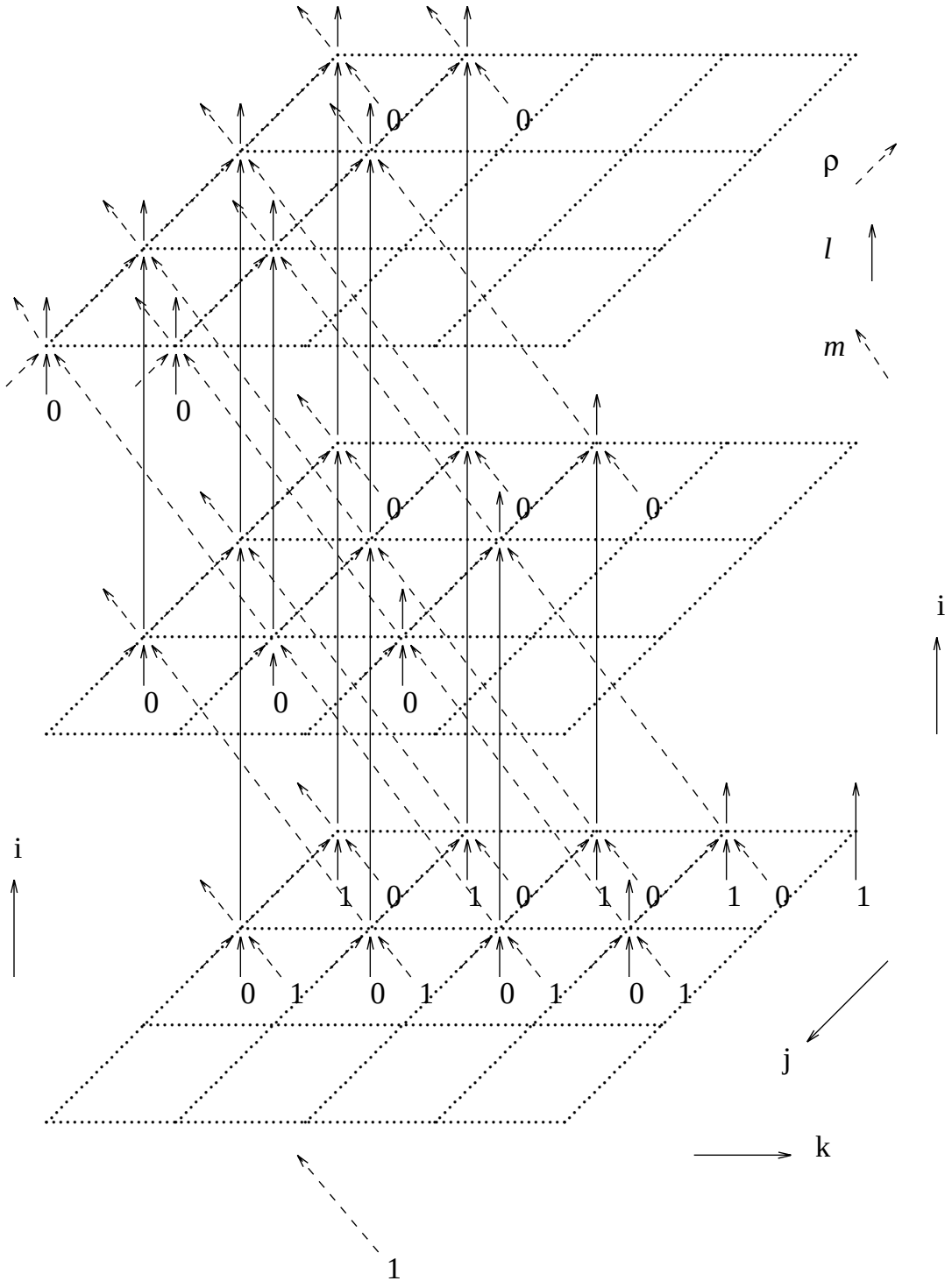
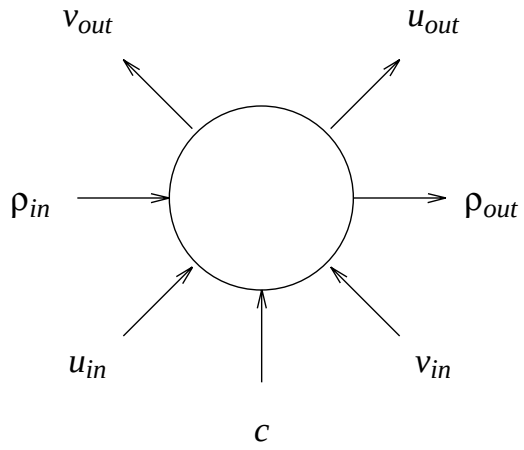


Figure 3.19. Dependence graph for computing the triangularization factors.



$$\textbf{if } c = 1 \quad \rho_{out} := -\frac{v_{in}}{u_{in}} \quad \textbf{else} \quad \rho_{out} := \rho_{in}$$

$$u_{out} := \frac{u_{in} + \rho_{out}\, v_{in}}{\sqrt{1 - |\rho_{out}|^2}}, \qquad v_{out} := \frac{v_{in} + \rho_{out}\, u_{in}}{\sqrt{1 - |\rho_{out}|^2}},$$

Figure 3.20. Processor that executes a basic operation of the Schur algorithm.

used by next iterations. Hence, next iterations can be computed in parallel with previous iterations, which means that they can be vectorized. Depending on the size of the vectors that can be processed by the computer, and depending on the amount of overhead that is introduced, several factors of speed-up may be achieved this way.

3.4.1.4 Incremental solution

To obtain an implementation of the Schur algorithm in which the approximate inverse is computed incrementally along the k direction and in which a significant reduction in memory usage is achieved, an alternative pair of dependence graphs for the Schur algorithm is presented in Figure 3.21 and Figure 3.22. The dependence graphs shown in Figure 3.21 and Figure 3.22 have been obtained from the dependence graphs given in Figure 3.18 and Figure 3.19 by mirroring the two latter around the $i,\ j$ plane and, in addition, renumbering the indices of the entries of G such that entry g_{pq} is swapped with entry $g_{N-q,N-p}$ (hence u_{pq} and v_{pq} are swapped with respectively entry $u_{N-q,N-p}$ and entry $v_{N-q,N-p}$). For the output matrices in Figure 3.22, L_{ME}^{-*} is now replaced by M_{ME}^{-*} and M_{ME}^{-1} is replaced by L_{ME}^{-1}, where - similar as in the previous case -

Algorithm 3.1. A vectorizable implementation of the Schur algorithm based on the dependence graphs shown in 3.18 and 3.19

```
for i := 0 ··· max(w)        # max(w) returns the maximum value in w[]
  for k := 1 ··· N
    if i ≤ w[k]
      if i ≠ 0
        ρ := - v[k][1] / u[k][1]          # compute reflection coefficient
      for j := 1 ··· w[k - j + 1] + 1 - i
        if i = 0
          u[k][j] := g[k - j + 1][j]                  # load U and V
          v[k - 1][j - 1] := g[k - j + 1][j]
        else
          u[k][j] := (u[k][j] + ρ * v[k][j]) / sqrt (1 - sqr (ρ))
          v[k - 1][j - 1] := (v[k][j] + ρ * u[k][j]) / sqrt (1 - sqr (ρ))
      for j := i + 1 ··· 1
        if i = 0
          l[k][j] = 1                             # load main diagonals of
          m[k - 1][j + 1] = 1                     # the identity matrices
        else
          if j = i + 1
            l[k][j] := 0                          # load zero entries of
          if j = 1                                # the identity matrices
            m[k][j] := 0
          l[k][j] := (l[k][j] + ρ * m[k][j]) / sqrt (1 - sqr (ρ))
          m[k - 1][j + 1] := (m[k][j] + ρ * l[k][j]) / sqrt (1 - sqr (ρ))
        if i = w[k]
          l̂[k][j] := l[k][j]                      # prepare L*_ME
        if k = 1 or i + 1 > w[k - 1]
          m̂[k - i][i - j + 2] := m[k - 1][j + 1]  # prepare M^-1_ME
```

$$G_{ME}^{-1} = L_{ME}^{-*}\, L_{ME}^{-1} = M_{ME}^{-1}\, M_{ME}^{-*}. \tag{3.75}$$

In Figure 3.21 the entries u_{pq} and v_{pq} of $U^{(n)}$ and $V^{(n)}$ after the i-th. upper diagonal of $V^{(n)}$ has been eliminated are output at respectively grid position $(i,\ q-p+1,\ p)$ and grid position $(i,\ q-p+1-i,\ p+i)$. In Figure 3.22 the entries m_{pq} and l_{pq} of M_{ME}^{-*} and L_{ME}^{-1} (M_{ME}^{-*} is lower triangular and L_{ME}^{-1} is upper triangular) after the i-th. upper diagonal of $V^{(n)}$ has been eliminated, are output at respectively grid position $(i,\ p-q+1,\ p)$ and grid position $(i,\ p-q+1+i,\ p+i)$.

An implementation of the Schur algorithm that is based on the dependence graphs shown in Figure 3.21 and 3.22 and that uses k as the main loop variable is given by Algorithm 3.2. This algorithm uses as input a matrix G that is stored identically as in Algorithm 3.1, it uses arrays $v[][]$ and $l[][]$ to store the intermediate results for $V^{(n)}$ and L_{ME}^{-1} for a particular value of k, and it computes an array $\hat{m}[][]$ for the entries of M_{ME}^{-*} such that entry $\hat{m}_{pq}$ of M_{ME}^{-*} corresponds to $\hat{m}[q][p-q+1]$.

From the dependence graphs shown in Figure 3.21 and Figure 3.22 it is possible to derive the following property for the Schur algorithm.

Property 3.4 :

Consider a normalized positive definite matrix G of size $N \times N$ that is specified on a staircase band $\{\ w_i \mid i = 1\ \cdots\ N\ \}$ (i.e. entry g_{ij} of G is unspecified for $j > i + w_i$ or $i > j + w_j$, and entry g_{ij} is specified in all other cases). Let the entries of G be only partially available, and let these entries be the entries g_{ij} for which $i \leq p$ or $j \leq p$. Then, with the Schur algorithm, the entries c_{ij} of G_{ME}^{-1} can be computed for which $i + w_i \leq p$ and $j + w_j \leq p$.

Proof:

Consider the matrix G in Figure 3.23a which has a support as indicated. Let the entries g_{ij} of G be available for which $i \leq p$ or $j \leq p$. As follows from Figure 3.21 and 3.22, the entries m_{ij} of M_{ME}^{-*} can then be computed for which $i \leq p$ (see Figure 3.23b), which means that all m_{ij}'s of M_{ME}^{-*} are known for which $j + w_j \leq p$. Since $G_{ME}^{-1} = M_{ME}^{-1} M_{ME}^{-*}$ and all m_{ij}'s of M_{ME}^{-1} are known for which $i + w_i \leq p$, the entries c_{ij} of G_{ME}^{-1} can be computed for which $i + w_i \leq p$ and $j + w_j \leq p$ (see Figure 3.23c)

□

Property 3.4 can be extended for an arbitrary positive definite matrix G in the

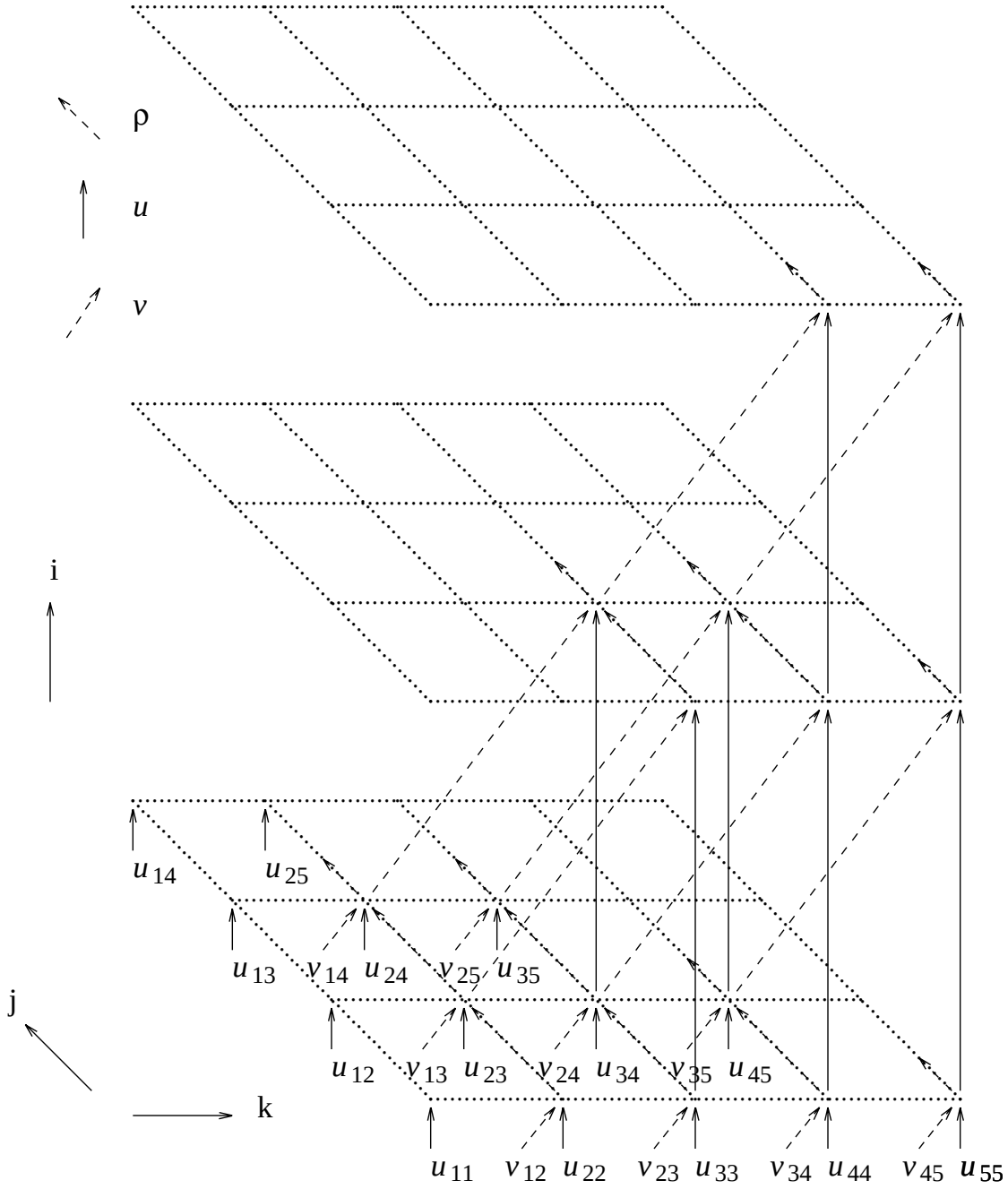


Figure 3.21. Alternative dependence graph for computing the reflection coefficients.

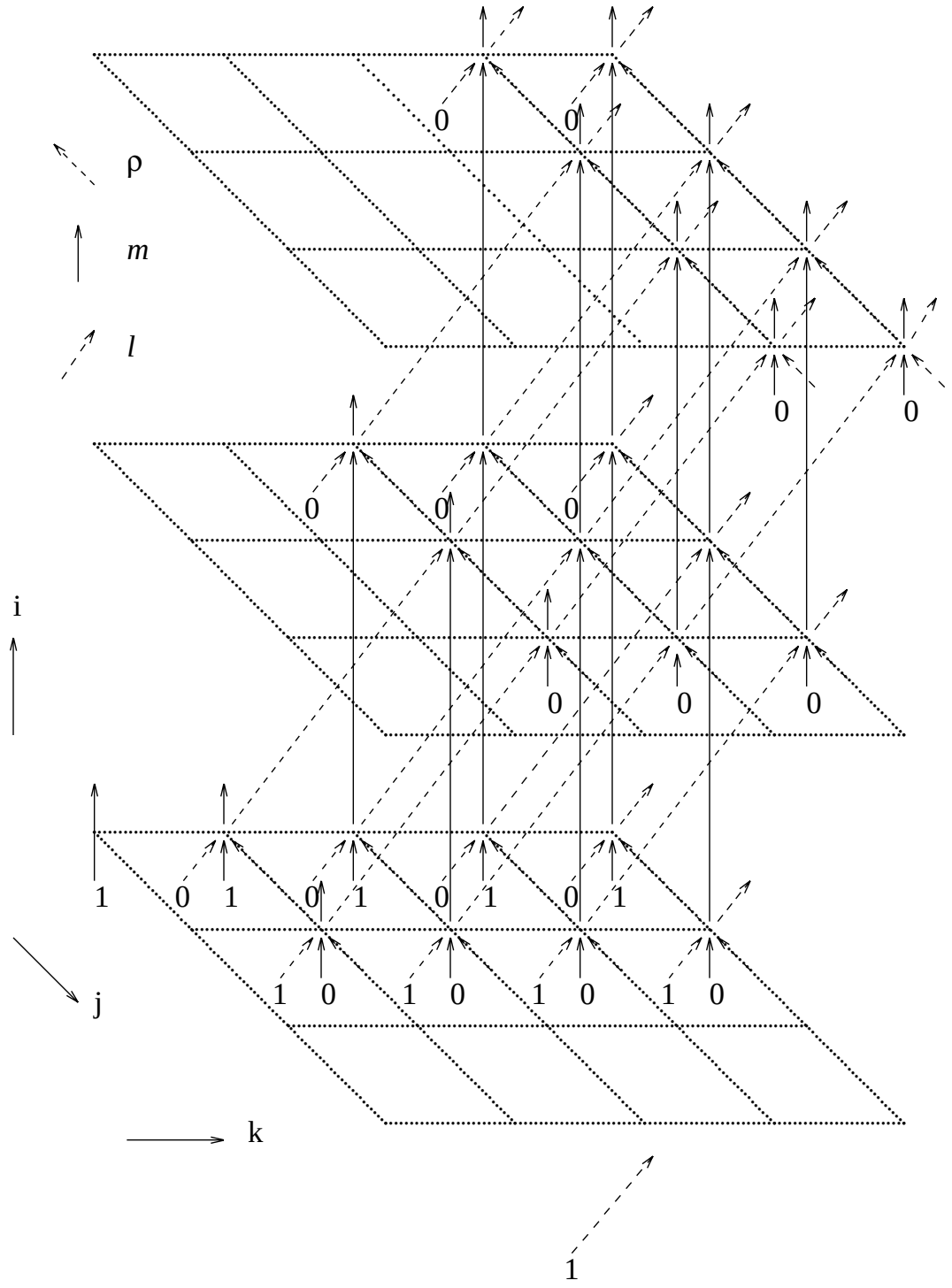
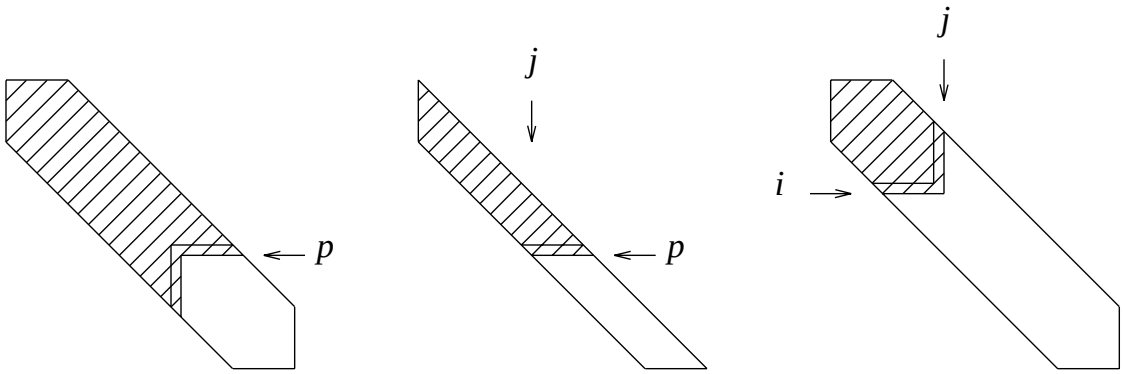


Figure 3.22. Alternative dependence graph for computing the triangularization factors.

Algorithm 3.2. An implementation of the Schur algorithm, based on the dependence graphs of 3.21 and 3.22, where the approximate inverse is incrementally computed from the input matrix.

```
for k := 1 ··· N
  max_i := 0
  for j := 1 ··· w[k] + 1
    i := 0
    while i < k and i ≤ w[k - i] + 1 - j
      if i = 0
        u := g[k][j]                          # load an entry of U and V
        v[j - 1][i + 1] := g[k][j]
      else
        if j = 1
          ρ[i] := - v[j][i] / u               # compute a reflection coefficient
        u := (u + ρ[i] * v[j][i]) / sqrt (1 - sqr (ρ[i]))
        v[j - 1][i + 1] := (v[j][i] + ρ[i] * u) / sqrt (1 - sqr (ρ[i]))
      max_i := max (max_i, i)
      i := i + 1
  for j := max_i + 1 ··· 1
    for i := j - 1 ··· max_i
      if i = 0
        m := 1                                # load an entry of the main
        l[j + 1][i + 1] := 1                  # diagonal of the identity matrix
      else
        if i = j - 1
          m := 0                              # load an entry of an off diagonal
        if j = 1                              # of the identity matrix
          l[j][i] := 0
        m := (m + ρ[i] * l[j][i]) / sqrt (1 - sqr (ρ[i]))
        l[j + 1][i + 1] := (l[j][i] + ρ[i] * m) / sqrt (1 - sqr (ρ[i]))
      m̂[k - j + 1][j] = m                     # prepare M̄*_ME
```



G M_{ME}^{-*} G_{ME}^{-1}

(a) (b) (c)

Figure 3.23. Scheme to solve M_{ME}^{-*} and G_{ME}^{-1} for a normalized positive definite matrix G that is specified on a staircase band: the dashed part of each of the matrices has been solved.

following way.

Property 3.5 :

Consider a positive definite matrix G of size $N \times N$ that is specified on a staircase band $\{ w_i \mid i = 1 \cdots N \}$ (i.e. entry g_{ij} of G is unspecified for $j > i + w_i$ or $i > j + w_j$, and entry g_{ij} is specified in all other cases). Let the entries of G be only partially available, and let these entries be the entries g_{ij} for which $i \leq p$ and $j \leq p$. Then, with the Schur algorithm, the entries c_{ij} of G_{ME}^{-1} can be computed for which $i + w_i \leq q$ and $j + w_j \leq q$, where $q + w_q \leq p$.

Proof:

Let the entries g_{ij} of G be available for which $i \leq p$ and $j \leq p$. The entries g_{ij} of G can then be normalized for $i \leq q$ or $j \leq q$, where $q + w_q \leq p$. Then, from property 3.4 it follows that the entries c_{ij} of G_{ME}^{-1} can be computed for which $i + w_i \leq q$ and $j + w_j \leq q$.

□

Algorithm 3.3 performs all steps of the approximate matrix inversion technique and uses property 3.5 to incrementally compute the approximate inverse. The algorithm assumes that the input matrix G becomes column-wise available, and it produces a new column of G_{ME}^{-1} as soon as this is possible according to property 3.5. The entries of G and G_{ME}^{-1} are stored in arrays $g[][]$ and $c[][]$ such that entry

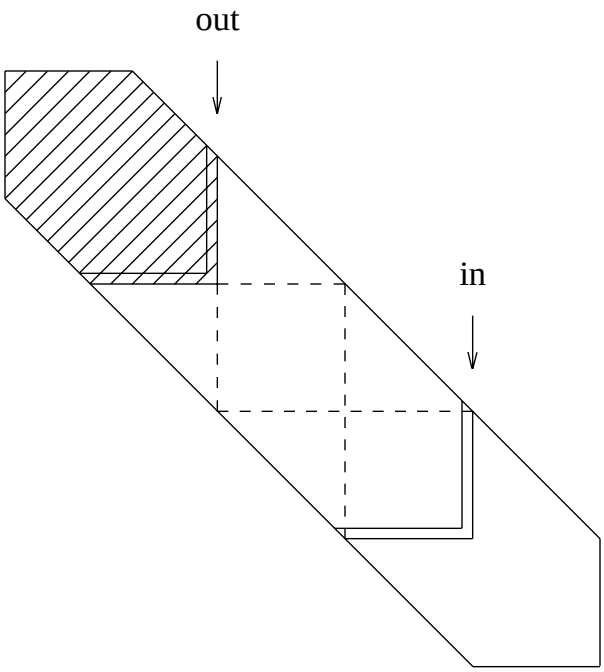


Figure 3.24. Scheme to solve G_{ME}^{-1} from an positive definite matrix G that is specified on a staircase band.

g_{pq} of G corresponds to $g[p][q-p+1]$, and such that entry c_{pq} of G_{ME}^{-1} corresponds to $c[p][q-p+1]$.

When b is the maximum bandwidth of G (i.e. the maximum value of w_i ($i = 1 \cdots N$)) it follows from the implementation of the Schur algorithm as given by Algorithm 3.2 and Algorithm 3.3 that the total memory that is needed by the Schur algorithm is dominated by

$\frac{1}{2}b^2 + \frac{3}{2}b + 1$ entries to store the part of the input matrix G that is being normalized,

$\frac{1}{2}b^2 + \frac{3}{2}b + 1$ entries to store the intermediate results of V,

$\frac{1}{2}b^2 + \frac{3}{2}b + 1$ entries to store the intermediate results of L_{ME}^{-1},

$\frac{3}{2}b^2 - \frac{5}{2}b + 1$ entries to store the part of M_{ME}^{-*} that is being used to compute G_{ME}^{-1}.

Hence, the Schur algorithm to find the approximate inverse of a matrix that has a maximum bandwidth b can be implemented to have a memory complexity that is

Algorithm 3.3. Algorithm to compute G_{ME}^{-1} for a positive definite matrix G that is specified on a staircase band.

```
k := 0
z := 0
for a := 1 ··· N                    # a counts the column of the upper triangular
                                    # part of G that becomes available
  while k + 1 + w[k + 1] ≤ a
    k := k + 1                      # k counts the row of the upper triangular part
                                    # of G that is ready to be processed
    execute one iteration of k of Algorithm 3.2,
    with g[k][j] replaced by g[k][j] / sqrt (g[k][1] * g[k + j][1])
    while z + 1 + w[z + 1] ≤ k
      z := z + 1                    # z is the column of the upper triangular part
                                    # of G_ME^-1 that becomes available
      x := z                        # x is used as a row index
      while x + w[x] ≥ z
                                    # compute entry c_xz of G_ME^-1 from M_ME^-1 M_ME^*
        c[x][z - x + 1] := 0
        for y := z - x + 1 ··· z - x + 1 + min (w[x] - z + x, w[z])
          c[x][z - x + 1] := c[x][z - x + 1] + m̂[x][y] * m̂[z][y - z + x]
        c[x][z - x + 1] := c[x][z - x + 1] / sqrt (g[x][1] * g[z][1])
        x := x - 1
```

$O(b^2)$. Further, from Algorithm 3.2, or the dependence graphs given in Figure 3.21 and Figure 3.22, it is easily derived that, when N is the size of the matrix that is inverted, the computation complexity of the Schur algorithm is $O(Nb^2)$.

3.4.2 Implementation of the extraction method

The boundary-element method and its approximate matrix inversion technique may be implemented by using a scanline technique [30, 31, 32]. In this technique, all steps of the method are executed as a scanline is moved over the layout of the circuit, say from left to right. When N is the size of the layout, measured in the number of mesh nodes that is generated, and when the distance over which

coupling effects are taken into account is constant, it can be shown that the method can be implemented to have a computation complexity that is $O(N)$ and a space complexity that is constant.

To describe the implementation of the extraction method based on the scanline technique, we consider a layout of height H and width W as shown in Figure 3.25. The maximum distance over which coupling effects are taken into account in the y direction is w_y, and the maximum distance over which coupling effects are taken into in the x direction varies between w_x and $2w_x$. As the scanline is swept over the layout from left to right, the conductors are recognized from the different mask layer combinations [32]. In order to compute the 3-D capacitances, the scanline is halted at positions $2w_x$, $3w_x$, $4w_x$, $\cdots$, $(M+1)w_x$, where $\frac{W}{w_x} > M \geq \frac{W}{w_x} - 1$. At each stop $2w_x$, $3w_x$, $4w_x$, $\cdots$, $(M+1)w_x$ of the scanline, a boundary-element mesh is constructed for the part of the circuit that is within a distance $2w_x$ to the left of the scanline. The corresponding boundary-element model for these strips is solved by using a numbering scheme and a window w_y in the y direction as shown in Figure 3.26. This corresponds to finding the contributions of 3.70a. At each stop $2w_x$, $3w_x$, $4w_x$, $\cdots$, Mw_x, of the scanline, a boundary-element mesh is constructed for the part of the circuit that is within a distance w_x to the left of the scanline. The corresponding boundary-element model for these strips is again solved by using a numbering scheme and a window w_y in the y direction as shown in Figure 3.26. This corresponds to finding the contributions of 3.70b.

For the implementation as described above, the pre- and post-multiplications performed to obtain the staircase band form for each of the sub-matrices are implicitedly executed by using a private numbering scheme for each of the strips. Further, if for each node in the mesh it is known to which conductor it corresponds, the contributions of the approximate inverses that are found for the sub-matrices can be added directly to the network capacitances and the (large) matrix G_{SI}^{-1} needs not to be maintained in memory. When c_{ij} is an entry of an approximate inverse that is calculated for a strip of width $2w_x$, and when node i is part of conductor I and node j is part of conductor J, then (see 3.1-3.4 and 3.70) the contribution of c_{ij} to the capacitances of the conductors is given by Algorithm 3.4, where C_I is the ground capacitance of conductor I, C_J is the ground capacitance of conductor J, and C_{IJ} is the coupling capacitance between conductor I and J. For a strip of width w_x an identical result is found, but now the reverse signed value of c_{ij} is used.

In order to obtain an estimate of the time and space complexity of the above implementation of the extraction method, we assume that the nodes of the

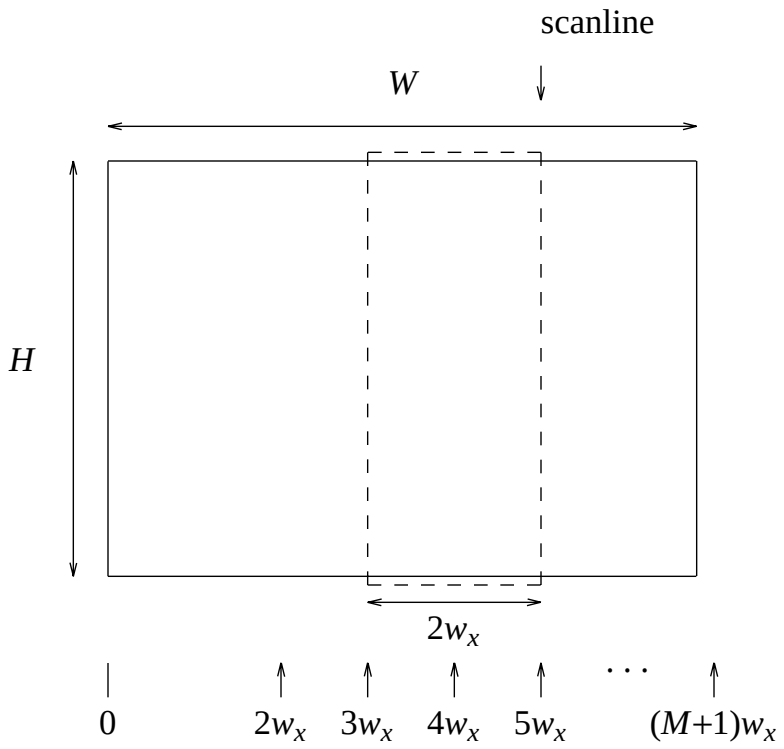


Figure 3.25. Stepping a window over the layout from left to right.

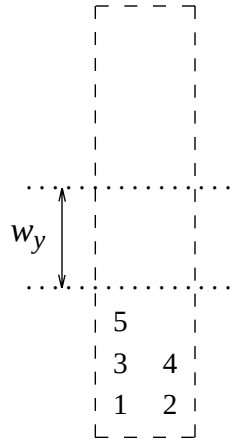


Figure 3.26. Numbering scheme and window for a strip that is found in Figure 3.25.

boundary-element mesh are uniformly distributed over the layout. As a result, for each strip, the size of the elastance matrix that is inverted is proportional to area of the strip (which is Hw_x) and the maximum bandwidth of the elastance matrix that is inverted is proportional to the area of the window (which is $w_y w_x$ or $2w_y w_x$). When using an incremental solution scheme as described in Section 3.4.1 it then

Algorithm 3.4. Algorithm to find the contribution of an entry c_{ij} of G_{ME}^{-1} to the conductor capacitances.

---

if $i = j$ # diagonal entry of the matrix

  $C_I := C_I + c_{ij}$

else if $I = J$ # nodes are on the same conductor

  $C_I := C_I + 2 * c_{ij}$

else # nodes are on different conductors

  $C_I := C_I + c_{ij}$
  $C_J := C_J + c_{ij}$
  $C_{IJ} := C_{IJ} - c_{ij}$

---

follows that the space complexity to find the approximate inverse for a single strip is $O(w_x^2 w_y^2)$. Since the strips are processed sequentially, the total space complexity for the approximate inversion method is therefore also $O(w_x^2 w_y^2)$. The computation complexity for a single strip is found according to the above results as $O(H w_x^3 w_y^2)$. Since there is a total number of $2M + 1$ different strips, where M is proportional to $\frac{W}{w_x}$, the total computation complexity for approximate inversion is therefore $O(HW w_x^2 w_y^2)$. When using $HW = O(N)$ and when assuming that the distances w_x and w_y are constant (i.e. independent of the size of the circuit) we then arrive at a total computation complexity for matrix inversion that is $O(N)$, and a total space complexity for matrix inversion that is constant.

To find the computation complexity for the complete extraction method we assume that the complete boundary-element mesh and the total set of elastance matrices are constructed with $O(N)$ computation complexity. This seems plausible since the complexity of the mesh and the total number of non-zero entries in the elastance matrices are proportional to the size of the circuit. Hence, in this case, the computation complexity of the complete extraction method is $O(N)$.

To find to the space complexity for the complete extraction method we consider two different cases. In the first case we assume that the boundary-element mesh and the elastance matrix are incrementally constructed and destroyed according to the position of the window. Then, a space complexity is found that is proportional to the size of the window and independent of the size of the circuit. However, the element mesh and the elastance matrix are created three times for most parts of the

circuit, which may cause a significant increase in computation time. In the second case we suppose that the mesh and the elastance matrix are maintained over a distance $2w_x$ to the left of the scanline. In this case no multiple creation of the mesh or the elastance matrix is required for some parts of the circuit, but now a space complexity for the complete extraction method is found that is $O(\sqrt{N})$.

In Table 3.4, Table 3.5 and Table 3.6 the CPU times and memory use are presented for the extraction of the interconnect capacitances in Figure 3.11, Figure 3.12 and Figure 3.16. The CPU times and memory use numbers that are given were obtained on an HP-9000/835 computer and they refer to the solution of the approximate inverse only.

Table 3.4. Capacitance values, CPU times and memory use on an HP-9000/835 when using the numbering scheme shown in Figure 3.11.

	capacitances (10^{-18} F)					CPU time	memory
w (μ)	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	(min:sec)	(kbyte)
10	1128	557.0	37.59	14.32	9.13	1:01.1	2323
7.75	1134	557.1	37.83	17.34	0	48.5	1859
5.5	1144	557.6	43.65	0	0	31.5	1393
3.25	1172	572.7	0	0	0	14.1	558
1	1519	0	0	0	0	2.8	93

Table 3.5. Capacitance values, CPU times and memory use on an HP-9000/835 when using the numbering scheme shown in Figure 3.12.

	capacitances (10^{-18} F)					CPU time	memory
w (μ)	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	(min:sec)	(kbyte)
10	1128	557.0	37.59	14.93	9.13	1:02.2	2279
8	1129	556.8	37.46	14.23	9.04	54.4	1478
6	1132	556.1	37.00	13.93	8.75	39.3	864
4	1142	554.1	35.85	13.27	8.19	22.9	864
3	1154	551.7	34.86	12.78	7.81	15.0	538
2	1180	547.4	33.68	12.25	7.37	8.9	228

In table 3.7, CPU times and memory use for matrix inversion are presented for different sizes of the circuit shown in Figure 3.16. The size of the circuit has been changed by increasing the length of the conductors and the number of conductors. For each circuit, an influence window was used that had an identical size. Beside the number of conductors, the length of the conductors, the CPU time and the memory use, Table 3.7 also shows for each circuit the number of nodes that was generated and the total number of non-zero entries in the elastance matrix that has

Table 3.6. Capacitance values, CPU times and memory use on an HP-9000/835 when using the numbering scheme shown in Figure 3.17.

	capacitances (10^{-18} F)					CPU time	memory
w_x, w_y (μ)	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	(min:sec)	(kbyte)
10	1128	557.0	37.59	14.93	9.13	1:00.6	2323
4.4	1148	554.1	36.06	15.87	0	18.4	553
2.2	1220	560.1	0	0	0	9.6	184

been computed (nr. of Greens). Note that the computation times for the different circuits are proportional to the size of the circuit, and note that the memory use for the different circuits is independent of the size of the circuit.

Table 3.7. CPU times and memory use on an HP-9000/835 for different sizes of the circuit when using a constant window (2μ∗2μ).

nr. of conductors	length (μ)	nr. of mesh nodes	nr. of Greens	CPU time (min:sec)	memory (kbyte)
5	10	220	12250	7	169
5	20	420	26150	16	169
10	20	840	55848	33	169
14	29	1680	122340	1:13	169
20	41	3360	253872	2:37	169

3.5 Application to extraction

The 3-D capacitance extraction method has been implemented in the layout-to-circuit extraction program SPACE [32, 33]. Besides performing extraction tasks such as transistor recognition, connectivity extraction, interconnect resistance calculation and interconnect capacitance calculation by means of parallel plate formulas [31], SPACE has the capability to optionally extract interconnect capacitances based on the 3-D capacitance model as described in Sections 3.2-3.4. In the current version of the program, any orthogonal three-dimensional conductor configuration is handled that is present in a stratified medium as shown in Figure 3.3. The coupling capacitances from the poly and metal wires to the diffusion wires are found by representing the diffusion wires as planar wires with an infinite small thickness that are present just above the substrate. Based upon the layout description and based upon on a parameter file that specifies the heights and thicknesses of the different conductors SPACE first generates a three-dimensional representation of the conductors, and then solves the 3-D capacitance model. The output of the program is a circuit description containing transistors, resistances

and/or capacitances that can directly be simulated by using a circuit simulator like SPICE. In order to find an optimal trade-off between accuracy and computation time, the user can adjust the fineness of the boundary-element mesh that is generated as well as the size of the influence window.

In Table 3.8 some extraction results are presented for a practical circuit design that was extracted by using SPACE. The circuit that was extracted is a CMOS static RAM cell developed in a 1 micron technology. The size of the influence window that was used was $3\mu * 3\mu$. The boundary-element mesh that was generated for this circuit is shown in 3.27.

Table 3.8. CMOS static RAM cell extraction statistics on an HP-9000/835.

total CPU time (min:sec)	10:41
time for matrix inversion	5:25
time for green evaluation	4:34
total memory (Mbyte)	6.9
# nr of mesh nodes	1550
# nr of Greens	411286

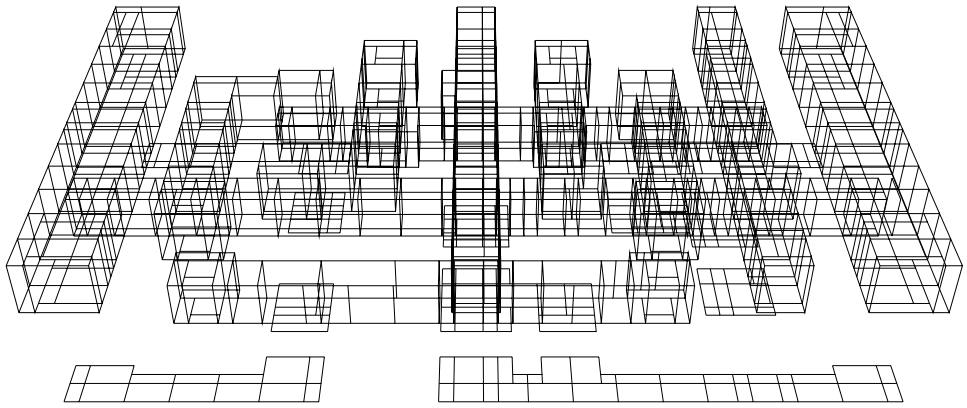


Figure 3.27. Mesh for a CMOS static RAM cell as generated by SPACE.

References

1. P.D. Patel, ‘‘Calculation of capacitance coefficients for a system of irregular finite conductors on a dielectric sheet,’’ *IEEE Trans. on Microwave Theory and Techniques* **MTT-19**(11) pp. 862-868 (Nov. 1971).

2. A.E. Ruehli and P.A. Brennan, ‘‘Efficient capacitance calculations for three-dimensional multiconductor systems,’’ *IEEE Trans. on Microwave Theory and Techniques* **MTT-21**(2) pp. 76-82 (Feb. 1973).

3. P. Benedek, ‘‘Capacitances of a planar multiconductor configuration on a dielectric substrate by a mixed order finite-element method,’’ *IEEE Trans. on Circuits and Systems* **CAS-23**(5) pp. 279-283 (May 1976).

4. A.E. Ruehli, ‘‘Survey of computer-aided electrical analysis of integrated circuit interconnections,’’ *IBM J. Res. Develop.* **23**(6) pp. 626-639 (Nov. 1979).

5. W.H. Dierking and J.D. Bastian, ‘‘VLSI parasitic capacitance determination by flux tubes,’’ *IEEE Circuits and Systems Magazine*, pp. 11-18 (Mar. 1982).

6. C.D. Taylor, G.N. Elkhouri, and T.E. Wade, ‘‘On the parasitic capacitances of multilevel parallel metallization lines,’’ *IEEE Trans. on Electron Devices* **ED-32**(11) pp. 2408-2414 (Nov. 1985).

7. P.E. Cottrell and E.M. Buturla, ‘‘VLSI Wiring Capacitance,’’ *IBM J. Res. Develop.* **29**(3) pp. 277-288 (May 1985).

8. Reinhold H. Uebbing and Masao Fukuma, ‘‘Process-Based Three-Dimensional Capacitance Simulation - TRICEPS,’’ *IEEE Trans. on CAD* **CAD-5**(1) pp. 215-220 (Jan. 1986).

9. F. Straker, *VLSICAP V1.3 User’s Guide*, Technische Universitat Wien, Wien, Austria (April 1986).

10. Z.Q. Ning, P.M. Dewilde, and F.L. Neerhoff, ‘‘Capacitance Coefficients for VLSI Multilevel Metallization Lines,’’ *IEEE Trans. on Electron Devices* **ED-34**(3) pp. 644-649 (March 1987).

11. A. Seidl, M. Svoboda, J. Oberndorfer, and W. Rosner, ‘‘CAPCAL - A 3-D Capacitance Solver for Support of CAD Systems,’’ *IEEE Trans. on CAD* **CAD-7**(5) pp. 549-556 (May 1988).

12. R. Guerrieri and A.L. Sangiovanni-Vincentelli, ‘‘Three-Dimensional Capacitance Evaluation on a Connection Machine,’’ *IEEE Trans. on Computer-Aided Design* **7**(11) pp. 1125-1133 (Nov. 1988).

13. Z.Q. Ning and P. Dewilde, ‘‘SPIDER: Capacitance Modelling for VLSI Interconnections,’’ *IEEE Trans. on Computer-Aided Design* **7**(12) pp. 1221-1228 (December 1988).

14. D.H. Norrie and G. de Vries, *The Finite Element Method*, Academic Press, New York (1973).

15. R.F. Harrington, ‘‘Matrix Methods for Field Problems,’’ *Proc. of the IEEE* **55**(2) pp. 136-149 (Feb. 1967).

16. Z.Q. Ning, ‘‘Accurate and Efficient Modeling of Global Circuit Behavior in VLSI Layouts,’’ Ph.D. Thesis, Delft University of Technology, Network Theory Section, Delft, the Netherlands (April 1989).

17. E. Weber, *Electromagnetic Fields, Theory and Applications*, John Wiley & Sons, Inc., New York (1957).

18. P. Silvester, ‘‘TEM wave properties of microstrip transmission lines,’’ *Proc. Inst. Elec. Eng.* **115** pp. 43-48 (Jan. 1968).

19. A. Ralston, *A First Course in Numerical Analysis*, McGraw-Hill , New York (1965).

20. P. Dewilde and E. Deprettere, ‘‘Approximate Inversion of Positive Matrices with Applications to Modelling,’’ *Modelling, Robustness and Sensitivity Reduction in Control Systems*, pp. 211-238 , Springer-Verlag (1987).

21. H. Nelis, E. Deprettere, and P. Dewilde, ‘‘Approximate Inversion of Positive Definite Matrices, Specified on a Multiple Band,’’ *Proc. SPIE 88*, San Diego, (Aug. 1988).

22. H. Nelis, ‘‘Sparse Approximations of Inverse Matrices,’’ Ph.D. Thesis, Delft University of Technology, Network Theory Section, Delft, The Netherlands (October 1989).

23. P. Dewilde, ‘‘New Algebraic Methods for Modelling Large-Scale Integrated Circuits,’’ *International Journal of Circuit Theory and Applications* **16** pp. 473-503 , John Wiley & Sons, Ltd (1988).

24. P. Dewilde and Z.Q. Ning, *Models For Large Integrated Circuits*, Kluwer Academic Publishers (1990).

25. S.Y. Kung, *VLSI Array Processors*, Prentice-Hall, Englewood Cliffs, N.J. (1988).

26. Jichun Bu, ‘‘Systematic Design of Regular VLSI Processor Arrays,’’ Ph.D. Thesis, Delft, the Netherlands (May, 1990).

27. A.J.J. de Lange, ‘‘Design and Implementation of Highly Parallel Pipelined VLSI Systems,’’ Ph.D. Thesis, Delft, the Netherlands (Jan. 24, 1991).

28. A.A.J. de Lange, A.J. van der Hoeven, E.F. Deprettere, and J. Bu, ‘‘An Optimal Floating-Point Pipeline CMOS CORDIC Processor: Algorithm, Automated Design, Layout and Performance,’’ *Proc. ISCAS-88*, Espoo, Finland, pp. 2043-2047 (June 7-9, 1988).

29. M.L.F. Lossie, ‘‘The Generalized Schur Algorithm: Roundoff Analysis, Vectorization, and an Application to VLSI Modelling,’’ Internal Report, Delft University of Technology, Network Theory Section, Delft, the Netherlands (Feb. 1988).

30. J.L. Bentley and T.A. Ottman, ‘‘Algorithms for reporting and counting geometric intersections,’’ *IEEE Trans. on Computers* **C-28**(9) pp. 643-647 (Sept. 1979).

31. N.P. van der Meijs and A.J. van Genderen, ‘‘An Efficient Algorithm for Analysis of Non-Orthogonal Layout,’’ *Proc. ISCAS-89*, pp. 47-52 (May 1989).

32. N.P. van der Meijs and A.J. van Genderen, ‘‘An Efficient Finite Element Method for Submicron IC Capacitance Extraction,’’ *Proc. 26th Design Automation Conference*, Las Vegas, pp. 678-681 (June 1989).

33. A.J. van Genderen and N.P. van der Meijs, ‘‘SPACE: A Finite Element Based Capacitance Extraction Program for Submicron Integrated Circuits,’’ *Software Tools for Process, Device and Circuit Modelling*, Dublin, Ireland, pp. 45-55 , Boole Press (July 1989).

4. SWITCH-LEVEL SIMULATION

4.1 Introduction

During the design process of integrated circuits, simulation is an important aid to verify the correctness of the design. Depending on the level of abstraction of the description of the circuit and depending on the simulation model that is used, important properties of the circuit can be verified, such as its electrical behavior, its timing behavior, its logical behavior, and its functional behavior. Although simulation does not prove that the circuit will function correctly under all circumstances, it allows the designer to study the combined influence of many different design parameters on the behavior of the circuit, and it gives him a clear impression of the characteristics of the circuit.

Different types of simulators can be distinguished, according to the type of circuit representation that is simulated, and according to the simulation model that is used. A circuit simulator such as SPICE [1] or ASTAP [2] uses as input the electrical representation of the circuit, including network details such as interconnect resistances and interconnect capacitances. Accurate transistors models and numerical techniques are used to compute precise values for voltages and currents as a function of time. In consequence, electrical properties can be studied such as voltage levels, power dissipation and dynamic behavior. Because of the use of much simulation detail, this type of simulator is only suited to simulate small and medium sized circuits.

Timing simulators [3, 4, 5, 6] use simplified transistor models, circuit partitioning techniques, and simplified numerical methods to compute voltage waveforms that have a lower accuracy than which can be obtained with a circuit simulator, but which still provide the necessary timing information. Further, these simulators directly simulate the transistor representation of a circuit, but they do not or only poorly model detailed electrical effects such as capacitive coupling. The advantage of this type of simulator over a circuit simulator is that also large circuits can be simulated within an acceptable period of time.

A gate-level simulator [7, 8] uses as input logic gates like NANDs and NORs. The nodes in the network are updated during simulation with a logic value like O, I or X. The timing of the circuit may in this case be simulated by assigning delay

times to the different signal transitions that occur in the circuit. Because of the use of simple event-driven Boolean evaluation techniques, these simulators are capable of simulating very large circuits in a short space of time. However, there is no direct relation between the electrical implementation of the circuit that is simulated, and many MOS-specific effects such as bidirectionality of signal flow and charge sharing are not well modeled.

Functional simulators [9, 10] describe the circuit in terms of building blocks such as registers, adders and multipliers. They are used in the initial phase of the design cycle in order to develop the architecture of the system and to verify the communication protocol of the circuit. These types of simulators efficiently simulate global circuit behavior, but they bear almost no relation to the transistor-level implementation of the circuit.

In order to combine the advantages of two, or more, of the above types of simulators, several other simulation models and techniques have been developed: the waveform relaxation technique [11] is used to significantly speed up simulation times for digital MOS circuits with almost no loss in accuracy; the piecewise linear simulation method [12] is applied to obtain a uniform modeling technique for circuit components that are described at a different level of abstraction. Another, important, type of such a simulator, which tries to combine some of the advantages of a circuit or timing simulator (accuracy, direct simulation of the transistor representation of the circuit) with the advantage of a gate-level or functional simulator (simulation speed), is a "switch-level simulator". A switch-level simulator [13, 14, 15] represents a MOS circuit as a set of nodes that are connected by transistor switches. It is capable of modeling many different MOS-specific effects such as bidirectionality of signal flow, dynamic charge storage and resistance division. By using event-driven simulation techniques, a simulation speed is obtained that is in the same order of magnitude as the speed of a gate-level simulator. Further, by using simple RC time constant evaluations [16] a first-order estimate of the timing behavior of the circuit can be obtained. Hence, the simulator can be used to directly and quickly simulate the transistor representation of a circuit, as for example obtained from a layout-to-circuit extraction, on its logic and timing behavior. However, a limitation of this type of simulator is that it is not as accurate as a circuit or timing simulator, and the simulator may fail to produce correct simulation results for digital circuits that are heavily dependent on their analog behavior (e.g. sense amplifiers and bootstrap drivers).

In this chapter, we extensively describe a switch-level simulator. First, in Section 4.2, we discuss the network model that is used by the switch-level simulator.

Second, in Section 4.3, we describe how the switch-level model is used to perform a logic simulation in which actual transistor and interconnection parameters are used to determine the logic values. Third, in Section 4.4, we show how a timing model that accurately represents transient effects like spikes and races is incorporated in the switch-level model. Next, in Section 4.5, there is a description of how function blocks are used in the network description, thus allowing mixed level simulation of circuits that are described at the transistor level, the gate level and the functional level, and making the simulator available for many stages in the design process. Finally, in Section 4.6, we treat a practical switch-level simulation program in which the above models and algorithms have been implemented.

4.2 Network model

4.2.1 Introduction

A important class of switch-level simulators [13, 17, 18] uses a MOS network model in which abstract values are used to model the transistor impedances as well as the node capacitances. For example in [18] each node has a size in the set $\kappa_1, \kappa_2, \cdots, \kappa_{max\kappa}$ and each transistor has a strength in the set $\gamma_1, \gamma_2, \cdots, \gamma_{max\gamma}$. The logic value of a node, which is O, I or X, is determined by the logic value of the input or supply node to which there is the lowest resistivity path, and/or the logic value of neighbor nodes that have the largest size. These simulation models can be implemented rather straightforwardly and they have led to speed enhancing extensions such as pre-compiling the network to some behavioral function prior to simulation [19, 20].

To provide more accurate simulation results, and to avoid the dubious classification of node capacitances and transistor impedances in sizes and strengths, other switch-level simulators have been developed that use network models that are based on linear transistor impedances and linear node capacitances [14, 21, 22, 23]. In these simulators, during simulation, linear node voltages are derived that are, by means of logic threshold voltages, converted to logic values O, I and X. With these simulators, it is possible to directly simulate the electrical netlist representation for a much wider variety of different circuits.

Here, we describe a switch-level network model that belongs to the second type [22, 24]. First, in Section 4.2.2, we give the transistor model and the interconnection model that are used by the simulator. Next, in Section 4.2.3 we show how the circuit is partitioned during simulation into "channel graphs" (or "vicinities") [18, 25]. A channel graph is a part of the circuit that can be evaluated separatedly from the rest of the circuit without modifying the result of the evaluation, and for which inputs and outputs can be indicated. By only evaluating

during simulation the channel graphs for which an input is changing, an efficient event-driven simulation is performed. Then, in Section 4.2.4, we introduce the network node state description that is used by the switch-level model. The network node state description that is introduced represents the signal at each node by a piece-wise linear minimum and maximum voltage waveform, and allows the accurate description of static effects like resistance division and charge sharing, and dynamic effects like spikes. Both a mininum and maximum voltage waveform are used used (1) to handle the logic value X during simulation, and (2) to support a min-max delay simulation. Finally, in Section 4.2.5, we give the simulation mechanism that is used by the simulator.

4.2.2 Transistor model and interconnection model

The electrical characteristics of a MOS transistor in a digital circuit, and in its normal "on" mode of operation, may be approximated by a linear impedance between drain and source, and capacitances attached to the gate, drain and source of the transistor [26, 16]. Using this information, a switch-level transistor model is introduced for a MOS transistor that is shown in figure 4.1. The model has a switch and a linear resistance in series between drain and source, and grounded capacitances to gate, drain and source. Accurate results of the transistor resistance and capacitances depend on the transistor type, the transistor dimensions used, the transistor function (e.g. pull-down, pass transistor), and on the type of evaluation that is performed (e.g. determining the stable voltages or determining the rise and fall times). The state of the transistor is Open, Closed or Undefined, according to the state of the switch. This state is derived from the logic value of the gate of the transistor according to Table 4.1.

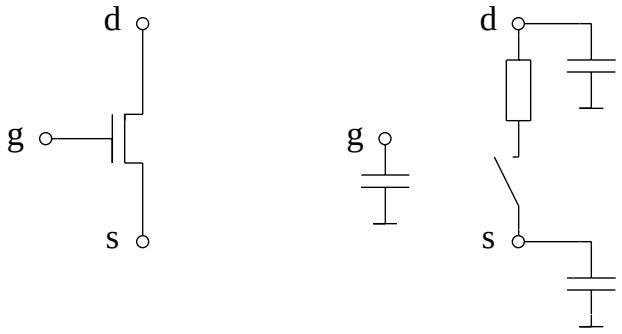


Figure 4.1. Transistor model.

The interconnections in the circuit are modeled by linear resistors and linear grounded capacitors. When the original circuit contains coupling capacitances these coupling capacitances are first converted into two ground capacitances that

Table 4.1. The logic value of the gate of a transistor and the corresponding transistor state for different types of transistors (nenh is an n-enhancement transistor, penh is a p-enhancement transistor and depl is a depletion transistor).

logic value of the gate	transistor state		
	nenh	penh	depl
O	Open	Closed	Closed
I	Closed	Open	Closed
X	Undefined	Undefined	Closed

are connected to the two nodes to which the coupling capacitance is connected, and that have a same value as the value of the coupling capacitance. This way the coupling effect between the two nodes is removed but - since the short-circuit capacitances of the circuit are unchanged - the timing properties of the circuit more or less remain the same.

4.2.3 Circuit partitioning

In order to efficiently simulate the behavior of the network by means of event-driven simulation techniques, the circuit is partitioned during simulation into "channel graphs". Before the definition of a channel graph is given, we first make a distinction between the different nodes that occur in the network.

Definition 4.1 :

A node in a MOS circuit that is directly connected to a voltage source (i.e. a supply node or an input node of the circuit), or to the ground potential, is said to be of a node of type Forced. The other nodes in the network are called nodes of type Normal.

A Forced node thus has a potential or logic value that cannot be influenced by neighbor nodes. Based on this property and based on the transistor model, the following definition of a channel graph (also called vicinity) is given.

Definition 4.2 :

A channel graph in a MOS circuit is a maximal set of nodes (vertices) connected by transistor drain-source channels and resistor elements (edges) such that each node in the graph is reachable from any other node in the graph by traversing drain-source channels of transistors that are Closed or Undefined and/or resistor elements, and such that no Forced node is crossed.

The partitioning of a circuit into channel graphs is illustrated in Figure 4.2. The channel graphs are bounded by the gates of the transistors, by the nodes of type

Forced, and by the drain and source connections of transistors that are Open. The partitioning of a circuit into its channel graphs dynamically changes during simulation as the transistors change their state.

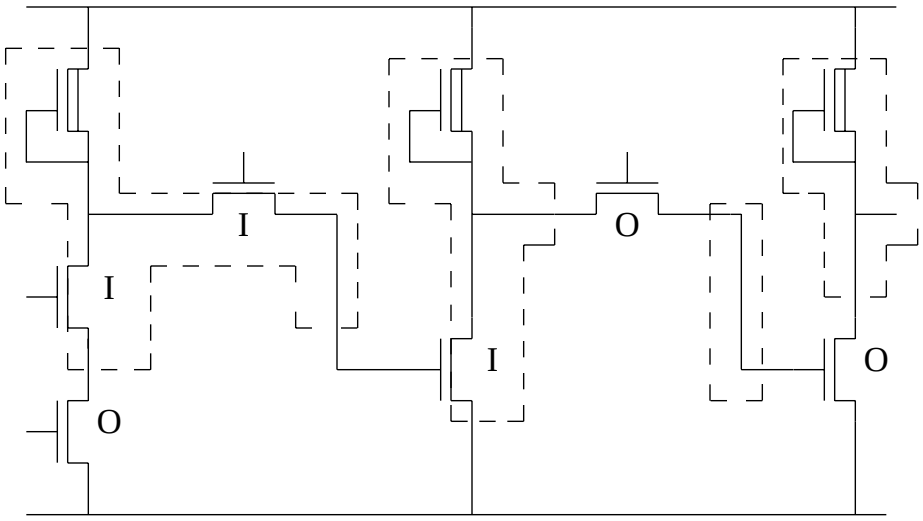


Figure 4.2. Example of partitioning a circuit into channel graphs. The channel graphs are denoted by dashed lines.

From the electrical model that was introduced in Section 4.2.2, it easily follows that each channel graph in the circuit is electrically isolated from the rest of the circuit. The only interaction with the rest of the circuit and the outside world takes place via the gates of the transistors of the graph and the Forced nodes of the graph (these are the inputs for the graph), and via the nodes of the graph that are connected to the gate of a transistor and the nodes of the graph that are an output of the circuit (these are the outputs of the graph). Hence each channel graph can be considered as a subcircuit that has inputs and outputs, and for which it is possible to determine the (future) values for the nodes in the graph from the values of the inputs and from the current values of the nodes in the graph. This leads to the following (evident) property for a channel graph.

Property 4.1 :

Consider a channel graph in a circuit at a time $t = t_1$. The channel graph is not disturbed from the outside, e.g. by the switching of a transistor that is part of the channel graph or that is adjacent to the channel graph, or by a value transition of a Forced node that is part of the graph, until a time $t_2 > t_1$. Then, from the values of the inputs of the channel graph and from the values of the initial voltages in the channel graph at $t = t_1$, it is possible to determine the voltage as a function of time for each node in the channel graph for $t_1 \leq t < t_2$.

This property is the basis of the definition of the state of a node, and for the event-driven simulation mechanism as employed by the simulator.

4.2.4 Signal representation

In most switch-level simulators the state of a node in the network is constituted by a logic value like O, I or X. This provides sufficient information to perform a logic simulation but it does not provide an accurate representation for transient effects like spikes and races. For example, when a node has just switched from O to I, the time to return to O will be much smaller than when the node has been I for a longer time. This effect cannot be modeled when using only a logic value like O, I or X for the state of a node. Therefore in [21] and [22] switch-level simulation models have been proposed in which voltage ramps are used to model the transition times for the logic value transitions in the circuit. Here, we will follow the model that is given in [22] which defines the state of the node to be given by a pair of linear splines that describe the current minimum and maximum voltage waveform for that node. Based on this information, it is possible, to store the amount of saturation for each signal, and to accurately compute the delay times also for the case where the signal has not yet stabilized. Thus, transient effects like spikes are more realistically modeled (see also Section 4.4).

A spline representation for the state of a node in the network is given by the vector

$$[\ iv_{\min},\ sv_{\min},\ Tt_{\min},\ tstab_{\min},\ iv_{\max},\ sv_{\max},\ Tt_{\max},\ tstab_{\max}\ ], \tag{4.1}$$

where $iv_{\min}$ and $iv_{\max}$ are the initial voltages of the minimum and maximum voltage waveform, $sv_{\min}$ and $sv_{\max}$ are the stable voltages of the minimum and maximum voltage waveform, $Tt_{\min}$ and $Tt_{\max}$ are the transition times of the minimum and maximum voltage waveform, and $tstab_{\min}$ and $tstab_{\max}$ are the stabilization times of the minimum and maximum voltage waveform. The above parameters describe a pair of linear splines $\{e_{\min}(t),\ e_{\max}(t)\}$ that are a function of time as follows (see also Figure 4.3)

$$e_{\min}(t) = \begin{cases} iv_{\min} & \text{for } t < tstab_{\min} - Tt_{\min} \\ iv_{\min} + (sv_{\min} - iv_{\min})\dfrac{t - tstab_{\min} - Tt_{\min}}{Tt_{\min}} & \\ & \text{for } tstab_{\min} - Tt_{\min} \le t < tstab_{\min} \\ sv_{\min} & \text{for } t \ge tstab_{\min} \end{cases} \tag{4.2a}$$

$$e_{\max}(t) = \begin{cases} iv_{\max} & \text{for } t < tstab_{\max} - Tt_{\max} \\ iv_{\max} + (sv_{\max} - iv_{\max})\dfrac{t - tstab_{\max} - Tt_{\max}}{Tt_{\max}} & \\ & \text{for } tstab_{\max} - Tt_{\max} \le t < tstab_{\max} \\ sv_{\max} & \text{for } t \ge tstab_{\max}. \end{cases} \quad (4.2b)$$

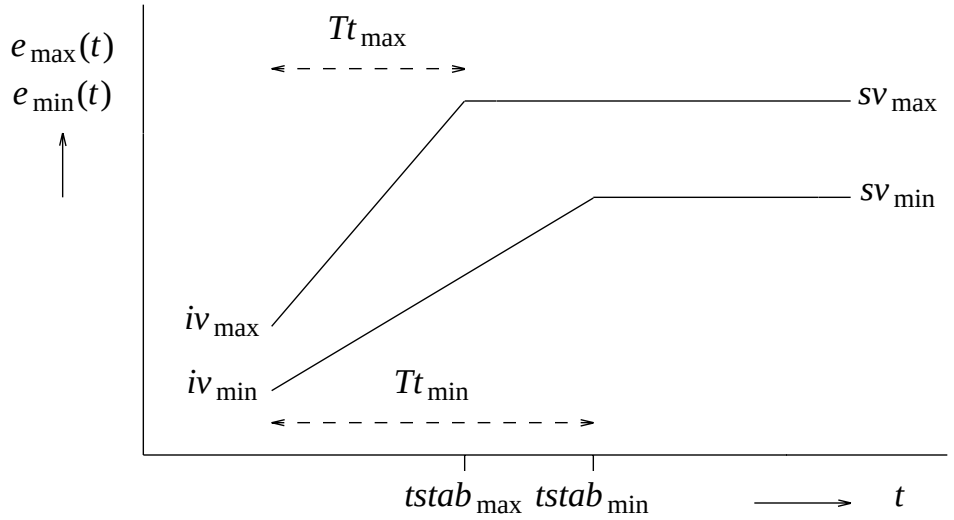


Figure 4.3. Splines functions $e_{\max}(t)$ and $e_{\min}(t)$ to represent the state of a node.

The idea is that the spline pair represents an estimation of the minimum and maximum voltage waveforms for the node, and that each time the node is evaluated a new pair of splines is computed that roughly bound the current rising or falling waveform for that node. According to Property 4.1, this result will be valid as long as the channel graph is not disturbed from the outside, e.g. by the switching of a transistor that is part of the channel graph or that is adjacent to the channel graph, or by a logic value transition of a Forced node that is part of the graph. The spline pairs are chosen such that the initial values $iv_{\min}$ and $iv_{\max}$ of a spline pair coincidence with the end values of the previous spline pair that was used for that node. Thus, the minimum voltage $u_{\min}(t)$ and the maximum voltage $u_{\max}(t)$ are continuous over the whole simulation time interval.

As an example of the representation of the minimum and maximum voltage waveform for a node, we consider the voltage waveforms that are shown in Figure 4.4. When $e^{(i)}_{\min}(t)$ and $e^{(i)}_{\max}(t)$ are the minimum and maximum spline that are computed for a particular node at time point t_i, where $i = 0, 1, 2, 3$, then the minimum voltage $u_{\min}(t)$ and the maximum voltage $u_{\max}(t)$ of the node are

represented by

$$u_{\min}(t) = \begin{cases} e^{(0)}_{\min}(t) \text{ for } t_0 \le t < t_1 \\ e^{(1)}_{\min}(t) \text{ for } t_1 \le t < t_2 \\ e^{(2)}_{\min}(t) \text{ for } t_2 \le t < t_3 \\ e^{(3)}_{\min}(t) \text{ for } t \ge t_3 \end{cases} \tag{4.3a}$$

$$u_{\max}(t) = \begin{cases} e^{(0)}_{\max}(t) \text{ for } t_0 \le t < t_1 \\ e^{(1)}_{\max}(t) \text{ for } t_1 \le t < t_2 \\ e^{(2)}_{\max}(t) \text{ for } t_2 \le t < t_3 \\ e^{(3)}_{\max}(t) \text{ for } t \ge t_3. \end{cases} \tag{4.3b}$$

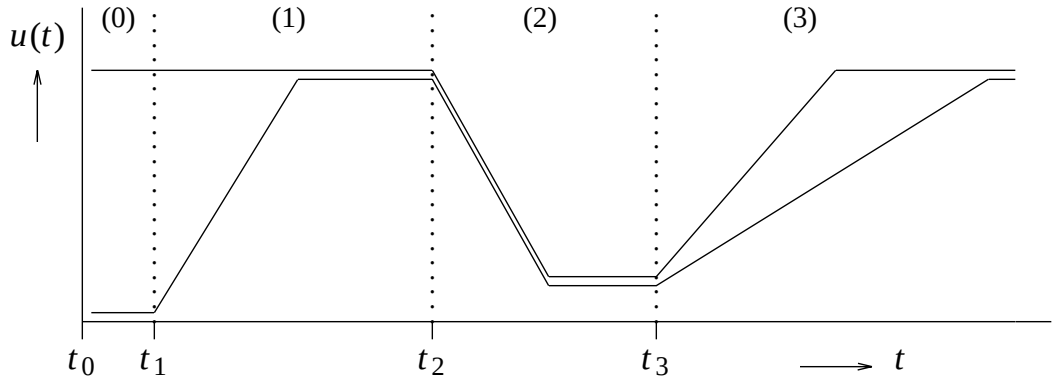


Figure 4.4. Voltage waveforms as formed by different spline representations.

In order to derive a logic level representation for the state of a node, and in order to find the state of the transistors that are connected to the node by their gate, we further define a logic value for each node. The logic value of a node is a Boolean value as a function of time which is either O (for low voltages), I (for high voltages) or X (for unknown). It is derived from the voltage waveforms by using a logic threshold voltage V_{switch}. For the moment - a final definition is given in Section 4.3.4 - we define the logic value *lvalue* of a node as

$$lvalue = \begin{cases} O & \text{if } u_{\max}(t) < V_{switch} \\ I & \text{if } u_{\min}(t) > V_{switch} \\ X & \text{otherwise.} \end{cases} \tag{4.4}$$

The voltage V_{switch} corresponds to the logic inversion voltage, and, in practice, for circuits the value of V_{switch} is approximately equal to half the value of the supply

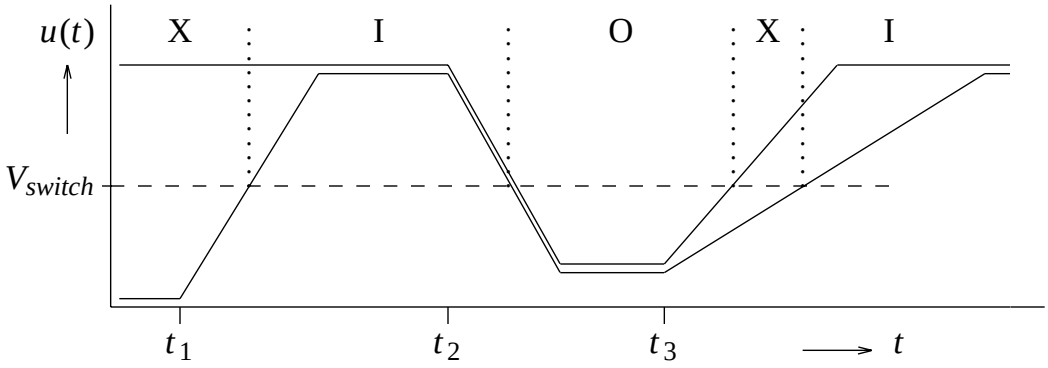


Figure 4.5. Logic value as a function of the node voltages.

voltage. As an example, Figure 4.5 shows the logic value as a function of time for the voltage waveforms shown in Figure 4.4. We will use this example to discuss some of the effects that are modeled by the state of the node. In the beginning the node is not yet initialized and the minimum and maximum waveforms are at the GND potential and the VDD potential respectively, so that the logic value is X. Then, at $t = t_1$, the state of the node is evaluated and a rising spline for the minimum voltage is computed. This causes the logic value of the node to change from X to I. Next, at $t = t_2$, both the minimum and maximum waveforms start falling and a logic value transition occurs from I to O, which is the normal situation for transient effects. For $t > t_3$, the logic value goes from O to I via the X value, which occurs because the minimum and maximum waveforms do not rise at the same speed. This may be caused by nodes that are in the neighborhood and that have not yet be initialized, or because the designer has introduced an uncertainty in the transition times (see Section 4.4.4).

The modeling of a spike with the linear splines is illustrated in Figure 4.6. At $t = t_1$, when the logic value of the node is O, a rising voltage waveform is computed for the node. The spline that is used to represent this rising voltage waveform is drawn by a solid line for $t_1 \leq t \leq t_2$ and by a dotted line for $t > t_2$. At $t = t_2$, when the logic value of the node has become I, and when the voltage waveform has not yet stabilized, an event occurs that causes to be a falling waveform computed for the node. Hence, the dotted part of the waveform does not become active to represent the voltage waveform for the node, but, instead, a spline is computed at $t = t_2$ that represents the (falling) voltage waveform for $t \geq t_2$. Note that the time to return from I to O after $t = t_2$ is much smaller than in the case where the voltage had stabilized at VDD and where it had fallen with the same slope.

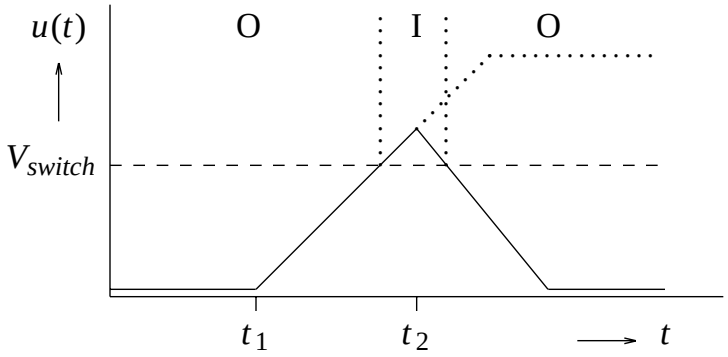


Figure 4.6. The modeling of a spike with linear splines.

4.2.5 Simulation mechanism

In this section we show how an event-driven simulation is performed using the switch-level model introduced in the previous sections. Since the simulation mechanism evaluates only channel graphs that are disturbed by a logic value transition, the simulation mechanism has a computation complexity that is linear with the number of logic value transitions in the circuit and, when the average size of the channel graphs in the circuit is not related to the size of the circuit, it is in principle independent of the size of the circuit.

A channel graph is evaluated when it is disturbed by the state transition of a transistor or by the value transition of a Forced node. The logic value transitions that cause these effects are called simulation events. As a channel graph is evaluated because a simulation event has occurred, new splines are computed for each node in the channel graph in order to obtain an estimate of the voltages in the channel graph (see Property 4.1). In consequence, new simulation events are possibly found that trigger the evaluation of other channel graphs. This has been illustrated in Figure 4.7. Suppose that at $t = 0$ the logic value at the gate of the n-enhancement transistor of the first inverter goes from I to O. This causes a simulation event to occur that requires the evaluation of the channel graph that is indicated in Figure 4.7a. The evaluation of this channel graph yields an event $O \rightarrow I$ for the input of the second inverter. Thus, the channel graph of the second inverter is evaluated and an event $I \rightarrow O$ is found for the output of the second inverter. This process continues until no new event is found and until no input signal changes its value.

An event δ may be characterized by a triplet $\{n, t, val\}$, where n is the node for which the event occurs, t is the time for which the event occurs, and val is the next logic value. To determine the sequence in which the events that have been computed become active, a "simulation eventlist" is maintained. The following

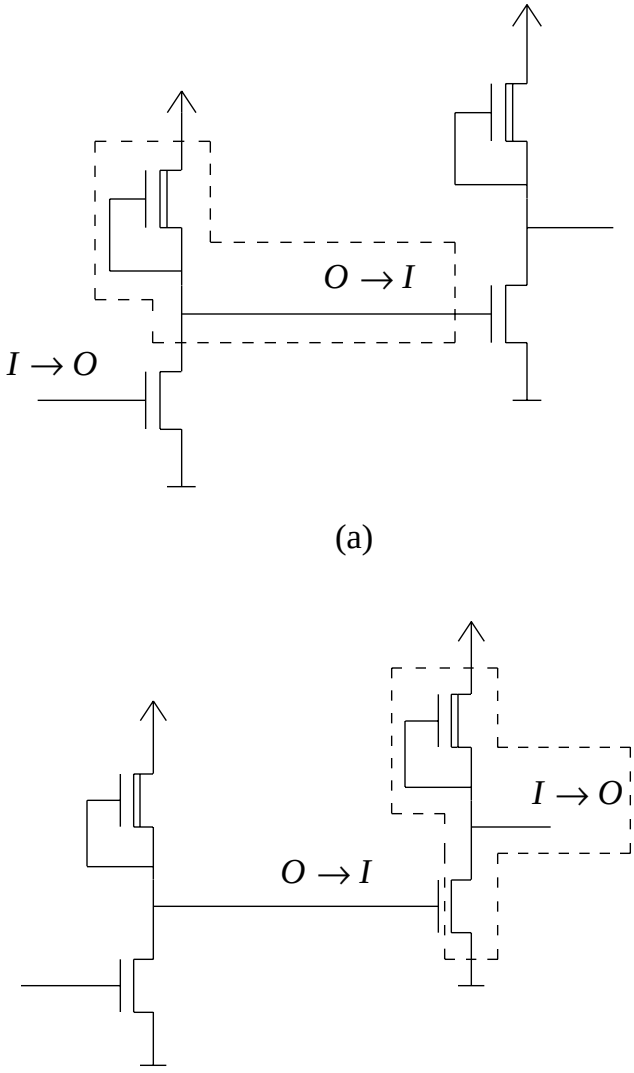


Figure 4.7. Simulation mechanism principle. The channel graph that is evaluated during each simulation step ((a) and (b)) is indicated by dashed lines.

operations should be possible for the eventlist:

schedule_event (δ)
: Schedule an event δ on the eventlist.

delete_event (*n*)
: Deletes a previously scheduled event for node *n* from the eventlist.

get_next_event ( )
: Fetches the next event from the eventlist and returns this event.

time_next_event ( )
Returns the time of occurrence of the next event.

The operations schedule_event () and delete_event () are performed during the evaluation of a channel graph. The eventlist can, for example, be implemented as a queue that contains the scheduled events sorted according to their time of occurrence.

In general, several simulation events may occur at the same time. Also, an event at $t = t_c$ may cause the scheduling of another event which occurs for the same time t_c, because of zero delay. To handle all these events in the correct order and to avoid conflicts, we use the following scheme to dispatch the events that have been scheduled on the eventlist for a particular time t_c: first all events are fetched from the eventlist that occur for $t = t_c$, the corresponding transistor states are updated, and the set of channel graphs that need to be evaluated is determined, second all the relevant channel graphs are evaluated. An action like this we will call a simulation step. Since first all channel graphs are determined that should be evaluated, and then all the evaluations take place, it is assured that within one simulation step a channel graph is evaluated at most once. Moreover, it is automatically guaranteed in this manner that each channel graph is evaluated with the same priority, and that the result of one evaluation of a channel graph does not influence the result of another evaluation of a channel graph that occurs within the same simulation step. The procedure to perform such a simulation step is given by Algorithm 4.1.

Algorithm 4.2 shows the procedure that executes all simulation steps that are necessary for a particular time t_c. To prevent the simulator from generating oscillations, the number of simulation steps that is executed for $t = t_c$ is limited by the logic depth of the circuit. Nodes that are still on the eventlist then for $t = t_c$ are put to X to indicate that their state is ambiguous.

In Algorithm 4.3 the main simulation loop is shown. After initialization of the node states and the transistor states, the eventlist is filled with the logic value transitions that occur at the network input nodes. Next, while recursively incrementing the current simulation time t_c to the time of the next event, the different simulation time steps are executed. As the channel graphs are evaluated new events will be scheduled on the eventlist and/or pending events will be deleted or rescheduled (because of a change in their time of occurrence or because of a change in the logic value for which they have been scheduled). The simulation stops when the eventlist is empty or when the simulation time t_c exceeds the maximum simulation time t_{stop}.

Algorithm 4.1. Procedure simulation_step.

```
procedure simulation_step ( )

  # S will contain a set of nodes that is such that of each channel graph
  # that needs to be evaluated at least one node is present in S.
  # It is assumed that initially n.evalflag = 0 for all nodes n.
  S := ∅

  while time_next_event ( ) = t_c
    δ := get_next_event ( )
    for all transistors t with t.gate = δ.n
      update t.state according to δ.val
      if t.drain.type = Normal S := S ∪ t.drain
      if t.source.type = Normal S := S ∪ t.source
    if δ.n.type = Forced
      for all transistors t with t.drain = δ.n and t.state ≠ Open
        if t.source.type = Normal S := S ∪ t.source
      for all transistors t with t.source = δ.n and t.state ≠ Open
        if t.drain.type = Normal S := S ∪ t.drain

  for all nodes n in S
    n.evalflag = 1

  for all nodes n in S
    if n.evalflag = 1
      search channel graph g of node n
      evaluate channel graph g
      for all nodes m in channel graph g
        m.evalflag := 0
```

4.3 Logic simulation

4.3.1 Introduction

In this section how a logic simulation is performed with the switch-level that was introduced in Section 4.2 is described. We give simulation algorithms that use actual transistor and interconnection parameters to determine the logic values and thus correct simulation results are obtained for many different types of digital MOS circuits.

Algorithm 4.2. Procedure current_time_step.

```
procedure current_time_step ( )

  step := 0
  while time_next_event ( ) = t_c and step ≤ logic_depth
    simulation_step ( )
    step := step + 1

  while time_next_event ( ) = t_c
    for all nodes n that are on the eventlist for t_c
      n.iv_min := n.sv_min := 0
      n.iv_max := n.iv_max := VDD
    simulation_step ( )
```

Algorithm 4.3. Procedure simulate

```
procedure simulate ( )

  for all nodes n
    n.iv_min := n.sv_min := 0
    n.iv_max := n.iv_max := VDD

  for all transistors t
    t.state := Undefined

  schedule events of network input nodes

  t_c := time_next_event ( )
  while t_c < t_stop
    current_time_step ( )
    t_c := time_next_event ( )
```

During logic simulation, new stable voltages $sv_{\min}$ and $sv_{\max}$ are computed for each node in a channel graph that is evaluated. To compute these stable voltages we use two different algorithms. In Section 4.3.2 how resistance division accounts for the effects that are caused by static currents that flow through the channel graph is described. In Section 4.3.3 charge-sharing algorithms are given that account for the redistribution of the initial charges in the channel graph. By

considering best-case and worst-case situations with respect to the state of the transistors and the state of the nodes in the channel graph, both algorithms yield a minimum and maximum value for the stable voltages for each node in the graph. These values are used to obtain the final minimum and maximum value for each node by taking the union of both intervals for each node. The algorithms that are given are efficient in the sense that their complexity is primarily determined by the size of the channel graph. To prevent the algorithms from having a nonlinear time complexity we have adopted methods that do no exhaustively test all possible input combinations to find the best-case and worst-case situations, but that provide a lower and upper bound in approximately linear time complexity. Finally Section 4.3.4 gives an extension of definition of the logic value of a node that is used to determine if the voltage level is sufficiently low or sufficiently high to form a valid logic O or a valid logic I.

4.3.2 Resistance division

For resistance division, the basic idea is to search for a minimum resistance path to Forced nodes with a logic value O, and a minimum resistance paths to Forced nodes with a logic value I, and then compute the stable voltages from a resistance division. This approach is motivated by (1) that the circuit is usually designed such that at least one (dominant) path to VDD or to GND is capable to bring each node to its correct logic value, (2) a path search algorithm to find the resistances to the Forced nodes will execute faster than e.g. solving the potentials from the set of simultaneous equations for the relation between the potentials and currents in the channel graph, (3) best-case and worst-case conditions are more straight forwardly handled for minimum path algorithms than for more general algorithms. Transistors that are parallel and that are both conducting may combined during a pre-pass on the channel graph in order to find an closer approximation of the stable voltages. To account for Forced nodes with a logic value X and for transistors with an Undefined state, best-case and worst-case minimum resistance paths are searched by the resistance division algorithm.

The resistance division algorithm is described in more detail as follows. First, for each node in the channel graph we search for (1) a best-case minimum resistance paths to Forced nodes with the logic value O, (2) a best-case minimum resistance path to Forced nodes with the logic value I, (3) a worst-case minimum resistance path to Forced nodes with the logic value O, and (4) a worst-case minimum resistance path to Forced nodes with the logic value I. The best-case and worst-case paths are found by searching through edges that correspond to Closed state transistors or resistors (worst-case), or to edges that correspond to Undefined and Closed state transistors or resistors (best-case), and by searching towards Forced

nodes with either only the logic value O or the logic value I (worst-case) or either the O and X value or the I and X value (best-case). For each different path, the resistances for all nodes in the graph may be found by applying a shortest path algorithm that starts at the target node(s) and then - by using a wavefront of last updated nodes - incrementally computes the minimum path resistance at each node [27, 28]. The results are summerized in Table 4.2. The minimum resistances for each path at each node will be denoted in the sequel by respectively bcr_O, bcr_I, wcr_O and wcr_I. When, at some node, a path is not connected to an appropriate Forced node, the resistance of the path at that point is given the value ∞. An example of the path resistances that are found for a channel graph is given by Figure 4.8.

Table 4.2. Minimum resistance paths that are searched for resistance division.

path	to Forced	through transistors
bc_O	O, X	Closed, Undefined
bc_I	I, X	Closed, Undefined
wc_O	O	Closed
wc_I	I	Closed

From Figure 4.8 we note that for some nodes the minimum paths to O and I may, over some distance, be running in parallel. This indicates that no static current is flowing through that part of the path and that its resistances should not be included in the resistance division. Therefore we assign to each node a "steering" node, which is the node where the actual resistance division for the node should take place. For a node n in a channel graph, its steering node n^{steer} is defined as follows.

Definition 4.3 :

Let n be a node in a channel graph that has best-case and worst-case mimum path resistances bc_O, bc_I, wc_O and wc_I according to the above definitions. If for node n $bcr_O < \infty$ and $bcr_I < \infty$, then the steering node n^{steer} is found by tracing bc_O and bc_I until the last node where they are together. Otherwise, the steering node n^{steer} is equal to n.

Let bcr_O, bcr_I, wcr_O and wcr_I be the path resistances at node node n, and let bcr_O^{steer}, bcr_I^{steer}, wcr_O^{steer} and wcr_I^{steer} be the path resistances at node n^{steer}. Then, the stable voltages for node n are computed from

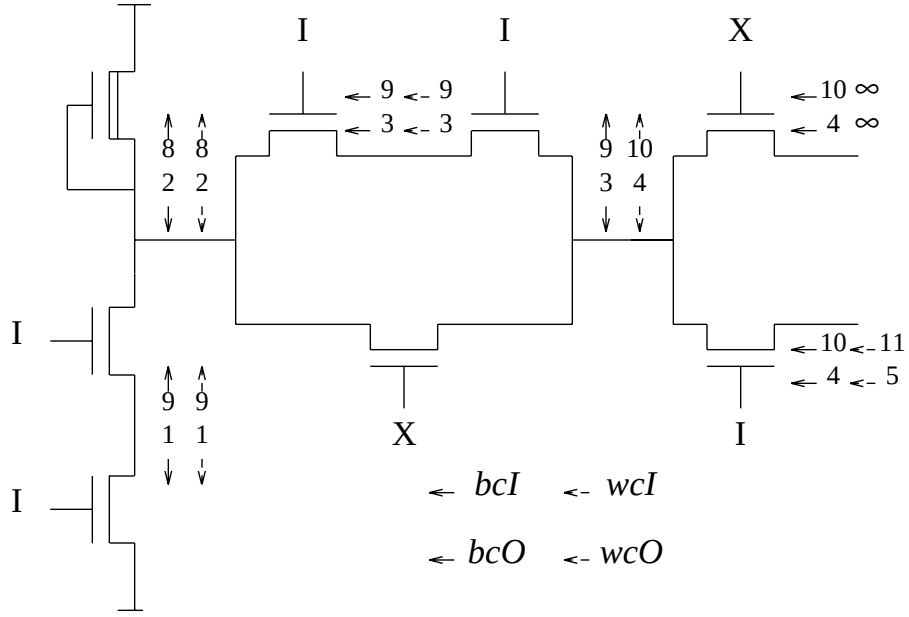


Figure 4.8. Example of the minimum path resistances that are found for a channel graph (the resistance of the depletion transistor is 8Ω and the resistances of the enhancement transistors are all 1Ω).

$$sv_{\min} = \begin{cases} VDD & \text{if } bcr_O = \infty \\ 0 & \text{if } bcr_O \neq \infty \text{ and } wcr_I = \infty \\ \dfrac{bcr_O^{steer}}{bcr_O^{steer} + wcr_I^{steer}} VDD & \text{otherwise} \end{cases} \tag{4.5a}$$

$$sv_{\max} = \begin{cases} 0 & \text{if } bcr_I = \infty \\ VDD & \text{if } bcr_I \neq \infty \text{ and } wcr_O = \infty \\ \dfrac{wcr_O^{steer}}{wcr_O^{steer} + bcr_I^{steer}} VDD & \text{otherwise.} \end{cases} \tag{4.5b}$$

As an example of the capabilities of the above resistance division algorithm we consider the CMOS EXOR circuit that is shown in Figure 4.9. The circuit has two inputs *in* 1 and *in* 2 and an output *out* that is intended to be high when either *in* 1 is high and *in* 2 is low, or when *in* 1 is low and *in* 2 is high. In all other cases *out* should be low. A switch-level simulator that uses abstract values to model the

transistor impedances has difficulties in simulating this circuit [29, 30]. When $in\,2 = I$ and $in\,1$ O $\rightarrow$ I, then a short-circuit path $in\,2$-*out*-$in\,1bar$-*GND* exists that causes a logic value X to be found for node $in\,1bar$ and node *out*. In contrast, the above resistance division algorithm finds that the potential of node $in\,1bar$ becomes low enough to be a logic O, so that $T\,6$ switches off and node *out* receives its correct value O.

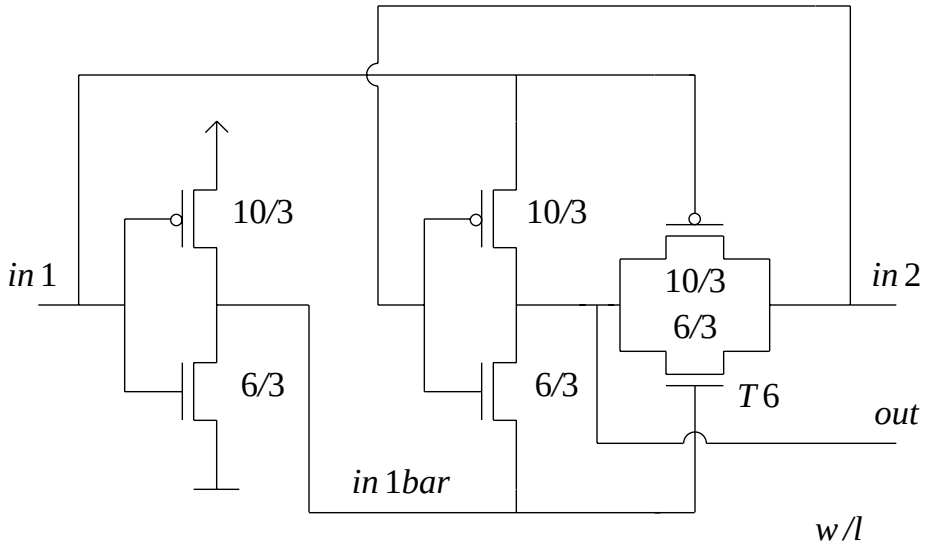


Figure 4.9. The Pass transistor EXOR gate.

Another example is given in Figure 4.10 which shows a selector-latch circuit that uses resistance division between a saturated and a nonsaturated transistor to obtain correct circuit behavior [31]. The functioning of the circuit is as follows: if cl = I then $out := \overline{in\,2}$; if cl = O then $out := (\overline{in\,1}$ or $out)$. This behavior is, among other things, based on the fact that when cl = I and $in\,2$ = O then b is O because $T\,4$ is saturated (hence its impedance is significantly higher than the impedance of transistor $T\,3$ although they have the same dimensions). The above resistance division algorithm will correctly simulate this circuit when applying an initial resistance division with normal impedances to detect that $T\,4$ is saturated, and then applying a final resistance division where an impedance is used for $T\,4$ that corresponds to its saturated mode.

4.3.3 Charge sharing

The charge-sharing effect becomes significant when the channel graph is not connected to a Forced node or when parts of it may be disconnected from Forced nodes because of Undefined state transistors. The basic idea behind computing the charge-sharings effect is simply to take the total charge of the relevant set of nodes

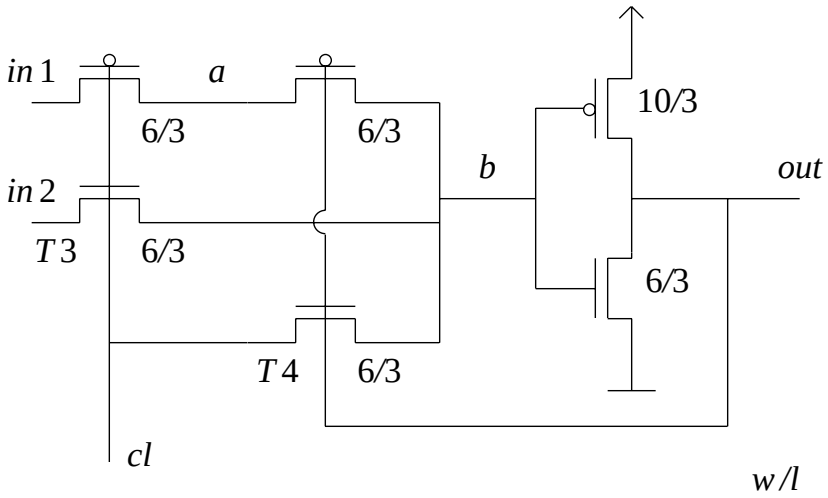


Figure 4.10. Selector-latch circuit.

and divide it by the total capacitance. Best-case and worst-case conditions are caused during the charge-sharing effect by minimum and maximum voltages, and by transistors with an Undefined state that partition the channel graph into groups of nodes.

To describe the charge-sharing algorithm in detail we first give the following definition.

Definition 4.4 :

A group in a channel graph is a maximal set of nodes connected by transistor drain-source channels and resistor elements such that each node in the group is reachable from any other node in the group by traversing drain-source channels of transistors that are Closed and/or resistor elements.

A group is thus a set of nodes that are always connected to each other and that can considered as a unit for charge-sharing evaluations. An example of the partitioning of a channel graph into its groups is given by Figure 4.11. From Figure 4.11 we see that some groups need not to be considered for the charge-sharing effect since they are connected to a Forced node via a resistor and/or drain-source channels of transistors that are in the Closed state. For example, in Figure 4.11, the stable voltages of the nodes of group 1 are not determined by a charge-sharing effect, but, instead, only by the resistance division effect. Hence these groups need not further to be considered for the charge-sharing effect. When removing these groups from the channel graph, the channel graph will be subdivided into agglomerations of groups between which there is no mutual

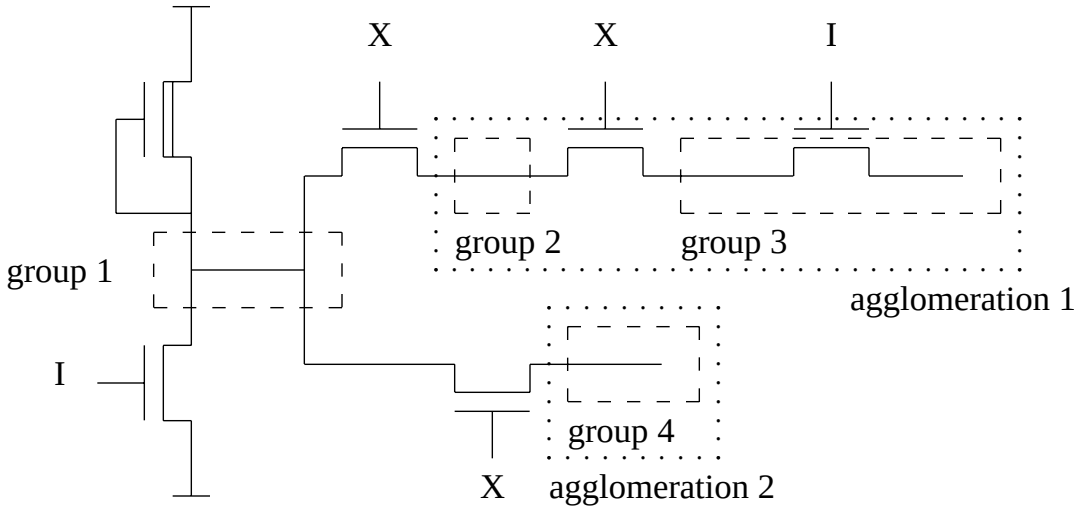


Figure 4.11. Partioning a channel graph into groups and agglomerations.

influence with respect to the charge-sharing effect.

Definition 4.5 :

Consider a channel graph that has been subdivided into groups. Let the groups that are directly connected to a Forced node via a Closed state transistor be removed from the channel graph. Then, an agglomeration is a maximal set of groups that are connected.

The actual charge-sharing algorithm is now applied to each agglomeration separatedly. First, minimum and maximum stable voltages are computed for each node for the case where the Undefined state transistors between the different groups of the agglomeration are all considered to be Open. The minimum and maximum stable voltages that are found for each group i in this case are

$$sv^1_{\min} = \frac{Q_{i,min}}{Cap_i} = \frac{\sum_{k \in g_i} C_k \, u_{k,min}(t_c)}{\sum_{k \in g_i} C_k} \tag{4.6a}$$

$$sv^1_{\max} = \frac{Q_{i,max}}{Cap_i} = \frac{\sum_{k \in g_i} C_k \, u_{k,max}(t_c)}{\sum_{k \in g_i} C_k}, \tag{4.6b}$$

where $Q_{i,min}$ is the minimum charge of group i, $Q_{i,max}$ is the maximum charge of group i, Cap_i is the capacitance of group i, t_c is the current simulation time, $u_{k,min}(t_c)$ and $u_{k,max}(t_c)$ are the minimum and maximum voltage of node k at time t_c, C_k is the capacitance of node k, and g_i is the set of nodes that belong to group i.

Based on this result, the groups are qualified as O groups, I groups and X groups according to

$$group \quad \begin{cases} I & if\ sv^1_{\min} > V_{switch} \\ O & if\ sv^1_{\max} < V_{switch} \\ X & otherwise. \end{cases} \tag{4.7}$$

Let *svOmin* be the minimum of the $sv^1_{\min}$'s of the O and X groups, *svImin* the minimum of the $sv^1_{\min}$'s of the I groups, *svOmax* the maximum of the $sv^1_{\max}$'s of the O groups, and *svImax* the maximum of the $sv^1_{\max}$'s of the I and X groups. Further, let the group capacitances other than the capacitance of group *i* be summed into Cap_X, Cap_O, and Cap_I, according the group type found in 4.7. Then, in linear time, a second upper bound and a lower bound for the voltages of group *i* are found from

$$sv^2_{\min} = minimum \{ \frac{Q_{i,min} + svOmin\ (Cap_O + Cap_X)}{Cap_i + Cap_O + Cap_X}, \frac{Q_{i,min} + svOmin\ (Cap_O + Cap_X) + svImin\ Cap_I}{Cap_i + Cap_O + Cap_X + Cap_I} \} \tag{4.8a}$$

$$sv^2_{\max} = maximum \{ \frac{Q_{i,max} + svImax\ (Cap_I + Cap_X)}{Cap_i + Cap_O + Cap_X}, \frac{Q_{i,max} + svImax\ (Cap_I + Cap_X) + svOmax\ Cap_O}{Cap_i + Cap_I + Cap_X + Cap_O} \}. \tag{4.8b}$$

The above values may further be adjusted to take into account the topology of the agglomeration. This is illustrated in Figure 4.12. The values of $\{iv_{\min}, iv_{\max}\}$ are shown for the three different groups that consist of node 1, 2 and 3 respectively. When applying 4.8, the combination of group 1 with the charge of the O group (group 3) provides $sv^2_{\min}$=1.67 for group 1. However, an exchange of charge between group 1 and group 3 will always be via group 2. Therefore, we make the following adjustment to the values as found in 4.8.

We say a group is Strong when both the $sv^2_{\min}$ and $sv^2_{\max}$ that are found for it, are below V_{switch}, or when they are both above V_{switch}. Let *svSmin* be the minimum of the $sv^2_{\min}$'s of the Strong groups and let *svSmax* be the maximum of the $sv^2_{\max}$'s of the Strong groups. Clearly, when there is more than one Strong group in an agglomeration, all Strong groups have a value O or they all have a value I. Also, when an I group is not connected with an O or X group other than via a Strong

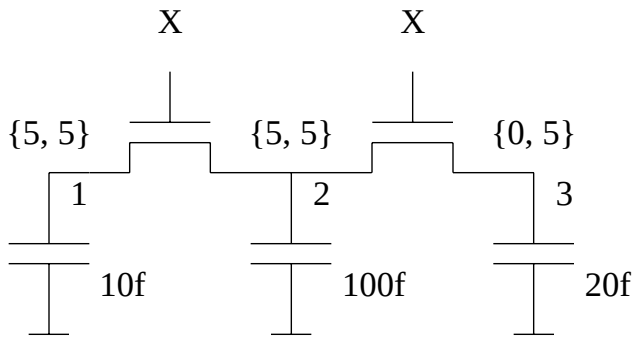


Figure 4.12. An example of an agglomeration consisting of three nodes (groups) where the initial voltages $\{iv_{\min}, iv_{\max}\}$ of the nodes are shown.

group with value I (or when an O group is not connected with an I or X group other than via a Strong group with value O) then the minimum (maximum) stable voltage of such a group will always be determined by a combination of only I groups (O groups) or by a combination of an arbitrary set of groups, with at least one group that is Strong. Therefore, when there are Strong groups in an agglomeration the following adjustments are made to the voltages that are found by 4.8.

1. Assign 'unprotected' to all groups that are not Strong and that have a value unequal to the value of the Strong groups, and assign 'protected' to all other groups.
2. Set the flag 'unprotected' for all groups that are not Strong and that are connected to a group that is 'unprotected' via groups that are not Strong.
3. If Strong groups have a value I, for groups that are 'protected' and that are not Strong, set $sv_{\min}$ to the minimum of *svImin* and *svSmin*. If the Strong groups have a value O, for groups that are 'protected' and that are not Strong, set $sv_{\max}$ to the maximum of *svOmax* and *svSmax*.

This will provide the final the stable voltages $sv^3_{\min}$ and $sv^3_{\max}$ for the charge-sharing effect. As an example, Table 4.3 shows the different voltages that are computed for the different groups in Figure 4.12.

4.3.4 A check on the voltage levels

According to the definition as given in Section 4.2, the threshold voltage V_{switch} forms an exact partitioning between the logic value O and the logic value I. However, the final stable voltages that are computed for a node may be found just above or just below V_{switch}. This, in general, does not correspond to a valid logic O or I. This situation is illustrated by the circuit that is shown in Figure 4.13,

Table 4.3. Stable voltages that are subsequently computed for the agglomeration in Figure 4.12.

	group		
	1	2	3
$sv^1_{\min}$, $sv^1_{\max}$	5, 5	5, 5	0, 5
$sv^2_{\min}$, $sv^2_{\max}$	1.67, 5	4, 5	0, 5
$sv^3_{\min}$, $sv^3_{\max}$	4, 5	4, 5	0, 5

which consists of an NMOS inverter for which a wrong ratio of the transistor impedances has been chosen. Therefore, the stable voltages of the node are computed as 2 Volt. When V_{switch} = 2.5 Volt, according to the definition in 4.4, this will produce an O as the final value. However, a voltage of 2 Volt will in practice not fully turn off an n-enhancement transistor that is connected to the node by its gate. Therefore, the logic value of the node is more appropriately qualified by an X. Similar situations occur, for example, with charge sharing when the driving capacitance is too small to fully charge or discharge connected nodes.

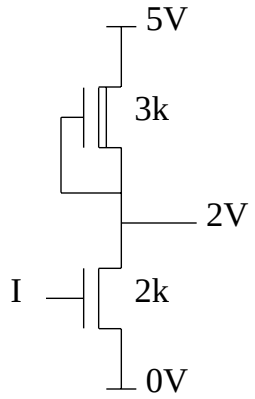


Figure 4.13. Inverter producing an invalid O because of wrong transistor ratios.

To prevent the simulator from generating invalid logic values we will introduce a minimum voltage for the logic value I, which is called V_{minI}, and a maximum voltage for the logic value O, which is called V_{maxO}. We then redefine the logic value *lvalue* of a node as

$$lvalue = \begin{cases} O & \text{if } u_{\max}(t) < V_{switch} \text{ and } (sv_{\max} \le V_{maxO} \text{ or } sv_{\min} > V_{switch}) \\ I & \text{if } u_{\min}(t) > V_{switch} \text{ and } (sv_{\min} \ge V_{minI} \text{ or } sv_{\max} < V_{switch}) \\ X & \text{otherwise.} \end{cases} \quad (4.9)$$

Practical values for V_{minI} and V_{maxO} are for example V_{minI} = 3.5 Volt and V_{maxO} = 1.5 Volt.

4.4 Timing simulation

4.4.1 Introduction

For networks that are simulated with a switch-level model, their timing behavior may be estimated by using RC constant evaluations [14, 16, 32, 33, 34, 35, 36, 37]. In [16, 32] a method was proposed where each subcircuit of a MOS circuit (channel graph) is represented by an RC tree network consisting of linear resistances, linear ground capacitances and a voltage source that is attached to the root of the tree network. A lower bound and an upper bound are computed for the voltage waveform of each output node in the network. Based upon this work, several timing models have been developed that allow one to find realistic delay times for signal transitions in MOS circuits [33, 36, 37]. The RC tree network method was later extended towards networks also containing resistance loops [34, 35, 38] and towards networks that are not connected to a voltage source but that are driven by charge-sharing effects only [39].

In the following we will describe how RC constant evaluations are used to find the transition times *Tt* and the stabilization times *tstab* for the switch-level model that was introduced in Section 4.2. Thereby, in order to accurately represent transient effects like spikes, the stable voltages of the nodes are made use of as well as the initial voltages of the nodes. An RC tree network model rather than a general RC network model is chosen since the switch-level model that was introduced in Section 4.2 deals with worst-case and best-case situations. For a tree network model, it is more or less possible to explicitedly determine for a transistor that has an Undefined state whether its Closed state or its Open state contributes to a best-case situation or a worst-case situation, while for a general network model this not possible without extensively trying all input combinations for all transistors that have an Undefined state. First, in Section 4.4.2, we examine how a first-order voltage waveform approximation is obtained for a node in an RC network. Second, in Section 4.4.3, how this result for an RC tree network is used to compute the timing parameters for the switch-level model that was introduced in Section 4.2 is described. Further, also in this section, a few examples are given that illustrate the accuracy of the timing model. Third, in Section 4.4.4, how a min-max delay simulation is performed with the switch-level timing model is depicted.

4.4.2 First-order waveform approximations in RC networks

Consider a linear RC network with nodes $1 \cdots N$, a ground capacitance C_k connected to each node k $\{ k \mid k = 1 \cdots N \}$, and a resistance R_{kl} between (some) node pairs k, l $\{ k, l \mid k = 1 \cdots N, l = 1 \cdots N, k \neq l \}$. At $t = 0$ the voltages of the nodes in the network may have a value different from zero. Our intention is to find an approximation for the voltage waveform $u_i(t)$ of each node i $\{ i \mid i = 1 \cdots N \}$ for $t \geq 0$ under the assumption that the network is not disturbed from the outside, and by using the spline representation that was introduced in Section 4.2.4. As conditions to find the "best approximating" spline $e_i(t)$ for the potential $u_i(t)$ of node i in the RC network we will use

$$e_i(0) = u_i(0) \tag{4.10a}$$

$$e_i(\infty) = u_i(\infty) \tag{4.10b}$$

$$\int_0^\infty [\, e_i(t) - u_i(t) \,]\, dt = 0. \tag{4.10c}$$

Further, we will use a zero inertial delay time for the approximating spline, hence

$$tstab_i = Tt_i. \tag{4.11}$$

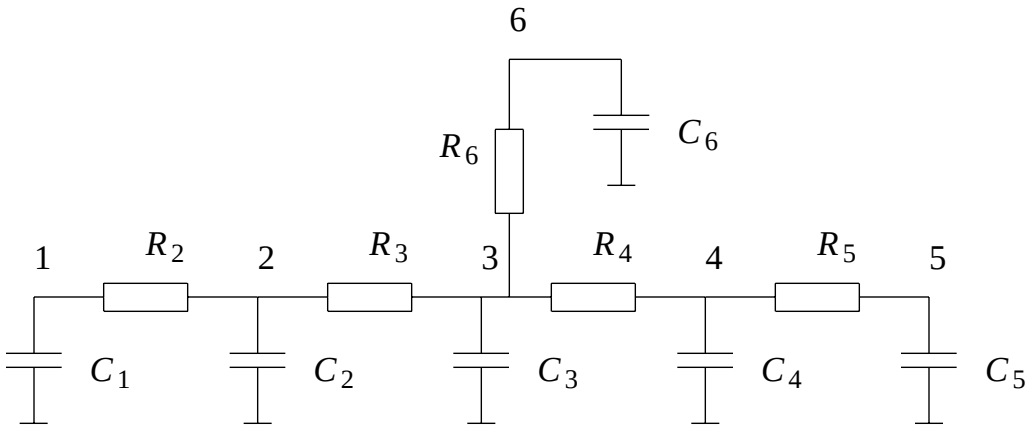


Figure 4.14. Example of an RC tree network.

In order to find the desired approximation we first give three definitions.

Definition 4.6 :

Consider a linear RC network with nodes $1 \cdots N$, a ground capacitance C_k connected to each node k $\{ k \mid k = 1 \cdots N \}$, and a resistance R_{kl} between (some) node pairs k, l $\{ k, l \mid k = 1 \cdots N, l = 1 \cdots N, k \neq l \}$. The potential as a function of time of each node i $\{ i \mid i = 1 \cdots N \}$ is given by $u_i(t)$. Then we define for each node i

$$T_{Di}^{\nu} = \int_0^{\infty} [\, u_i(\infty) - u_i(t) \,]\, dt. \tag{4.12}$$

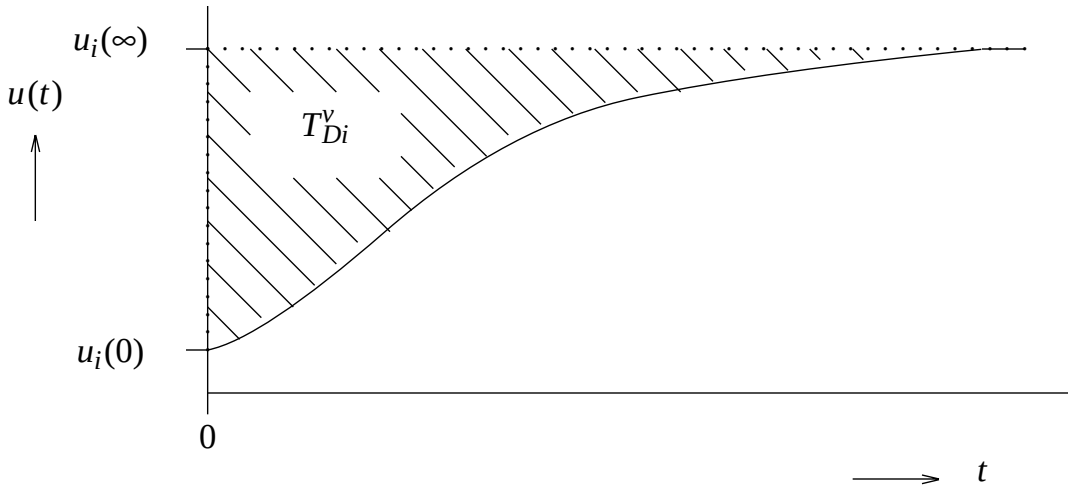


Figure 4.15. Interpretation of T_{Di}^{ν}.

Definition 4.7 :

Consider a linear resistance network with nodes $1 \cdots N$, in which there is at least one path via resistances between two nodes of the network, and let $u_1 \cdots u_N$ represent the voltages of the nodes in the network and $i_1 \cdots i_N$ the currents that are flowing into the nodes of the resistance network. Then, R_{kij} for each node triplet k, i and j $\{ k, i, j \mid k = 1 \cdots N, i = 1 \cdots N, j = 1 \cdots N \}$ is defined as

$$R_{kij} = \frac{u_i - u_j}{i_k} \qquad i_l = 0,\ i_k \neq 0,\ (l = 1 \cdots N, l \neq k). \tag{4.13}$$

Definition 4.8 :

Consider a linear RC network with nodes $1 \cdots N$, a ground capacitance C_k connected to each node k $\{ k \mid k = 1 \cdots N \}$, and a resistance R_{kl} between (some) node pairs k, l $\{ k, l \mid k = 1 \cdots N, l = 1 \cdots N, k \neq l \}$. The potential as a function of time of each node k $\{ k \mid k = 1 \cdots N \}$, is given by $u_k(t)$. Then we define for each node pair i, j

$$\tau_{ij}^{\nu} = \sum_{k=1}^{N} R_{kij}\, C_k\, (\, u_k(\infty) - u_k(0)\,), \tag{4.14}$$

where R_{kij} is given by definition 4.7.

As an example of the interpretation of T^{ν}_{Di}, see Figure 4.15. When $u_i(0) = 0$ ($i = 1 \cdots N$), and when the $u_i(t)$'s are normalized such that $u_i(\infty) = 1$ ($i = 1 \cdots N$), then T^{ν}_{Di} is equal to a value that is known as Elmore's delay [34, 40]. For an RC tree network, the value of R_{kij} is given by the resistance of the portion of the (unique) path between the node j and node i, that is common with the (unique) path between the node j and node k. For example, in Figure 4.14 $R_{4,6,1} = R_2 + R_3$. The values of T^{ν}_{Di} and T^{ν}_{Di} of a node pair i, j relate to the value of τ^{ν}_{ij} for this node pair in the following way.

Theorem 4.2 :

Consider a linear RC network with nodes $1 \cdots N$, a ground capacitance C_k connected to each node k $\{ k \mid k = 1 \cdots N\}$, and a resistance R_{kl} between (some) node pairs k, l $\{ k, l \mid k = 1 \cdots N, l = 1 \cdots N, k \neq l \}$. Then it holds that

$$T^{\nu}_{Di} - T^{\nu}_{Dj} = \tau^{\nu}_{ij}, \tag{4.15}$$

where T^{ν}_{Di} and T^{ν}_{Dj} are given by Definition 4.6 and τ^{ν}_{ij} is given by Definition 4.8.

Proof:

Since for two nodes i and j in the RC network it holds that $u_i(\infty) = u_j(\infty)$, we may write

$$T^{\nu}_{Di} - T^{\nu}_{Dj} = \int_0^{\infty} [u_j(t) - u_i(t)]\, dt. \tag{4.16}$$

With i_k being the current that is provided by capacitance C_k, the right-hand side of 4.16 can be written as

$$\begin{aligned}
\int_0^{\infty} [u_j(t) - u_i(t)]\, dt &= -\int_0^{\infty} \sum_{k=1}^{N} R_{kij}\, i_k \\
&= \int_0^{\infty} \sum_{k=1}^{N} R_{kij}\, C_k \frac{du_k(t)}{dt} dt \\
&= \sum_{k=1}^{N} R_{kij}\, C_k\, (u_k(\infty) - u_k(0)) \\
&= \tau^{\nu}_{ij}
\end{aligned} \tag{4.17}$$

(see Figure 4.16).

□

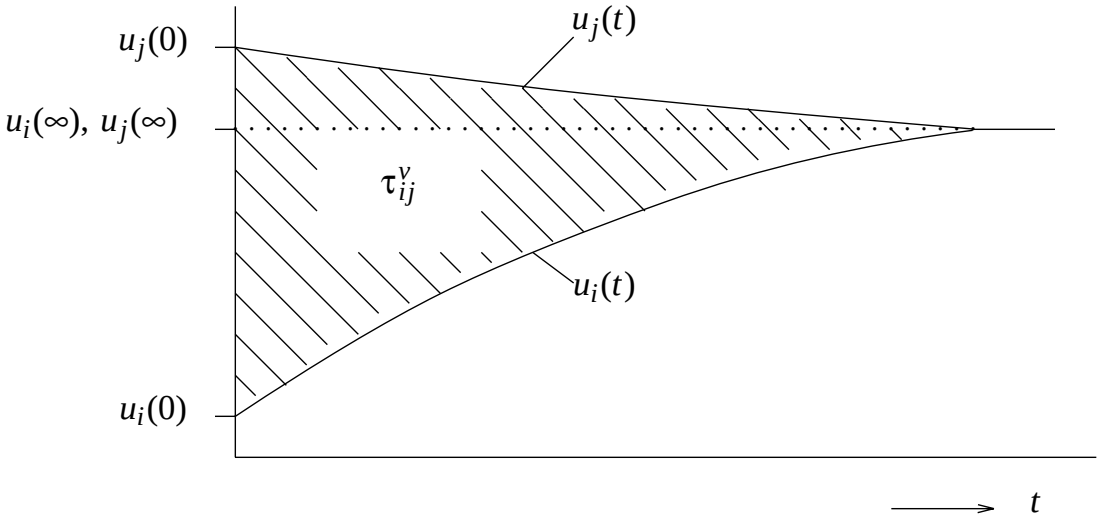


Figure 4.16. Interpretation of τ_{ij}^v.

To model channel graphs that are connected to a Forced node and channel graphs that are not connected a Forced node, we will in the following consider RC networks that are driven by a voltage source, and RC networks that are not connected to a voltage source.

4.4.2.1 RC networks driven by a voltage source

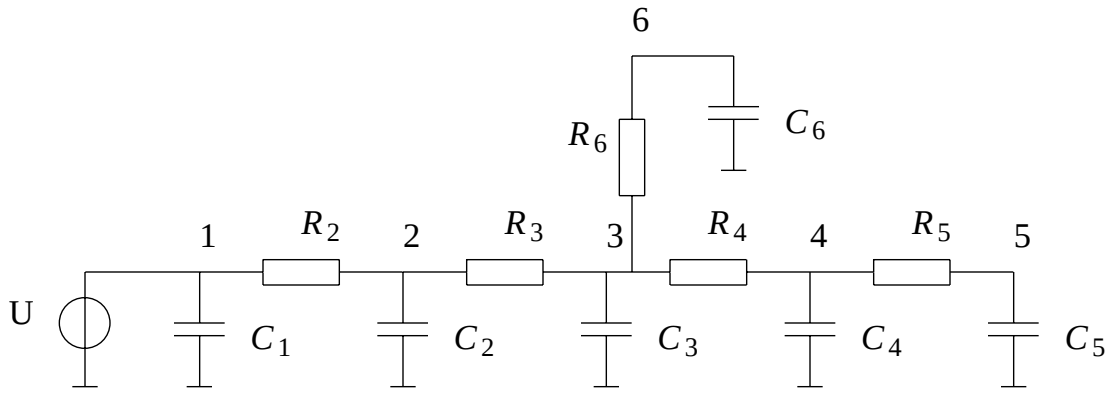


Figure 4.17. RC network driven by a voltage source.

For the case where the RC network is connected to a voltage source we assume that a voltage source with a constant potential U is connected to node j of the network. Then it follows that

$$T^{v}_{Dj} = 0. \tag{4.18}$$

Hence from Theorem 4.2 we find

$$T^{v}_{Di} = \tau^{v}_{ij}. \tag{4.19}$$

The boundary conditions 4.10a and 4.10b are satisfied by selecting

$$iv_i = u_i(0) \tag{4.20a}$$
$$sv_i = u_i(\infty) = U. \tag{4.20b}$$

This results in

$$\int_0^\infty [\, e_i(t) - u_i(t) \,]\, dt = \int_0^\infty [\, U - u_i(t) \,]\, dt - \int_0^\infty [\, U - e_i(t) \,]\, dt$$
$$= \tau^{v}_{ij} - \tfrac{1}{2} Tt_i \,(\, U - u_i(0) \,). \tag{4.21}$$

Thus, if we choose

$$Tt_i = \frac{2\,\tau^{v}_{ij}}{U - u_i(0)}, \tag{4.22}$$

we have found an approximation $e_i(t)$ for $t \geq 0$ for node i that satisfies all three conditions 4.10a, 4.10b and 4.10c (see Figure 4.18).

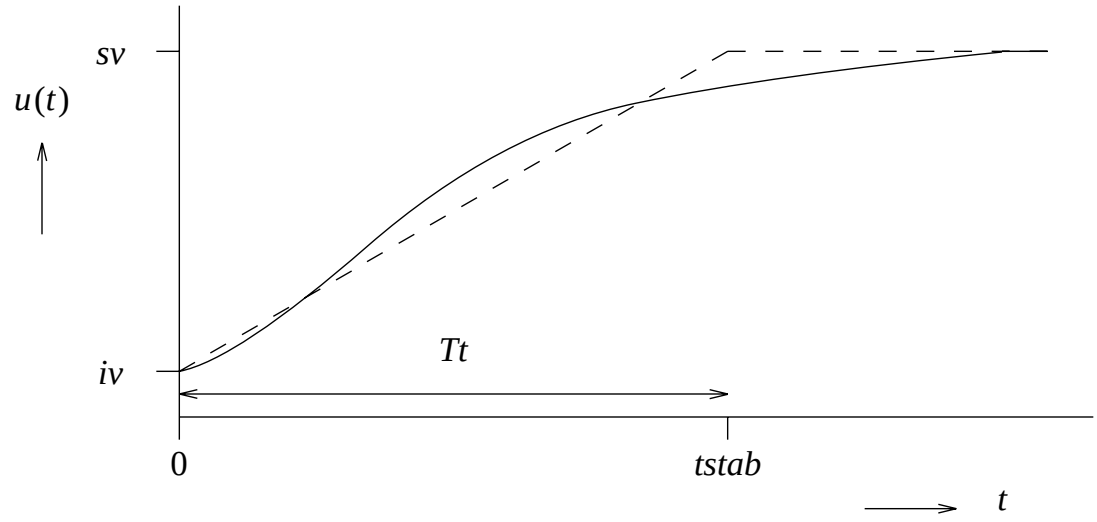


Figure 4.18. Example of a voltage waveform (solid line), and its spline approximation (dashed line).

4.4.2.2 RC networks not connected to a voltage source

To find the appropriate transition times for RC networks that are not connected to a voltage source we choose one of the nodes - say node r - as a reference node. For the final voltages of the nodes in the RC network we have

$$u_i(\infty) = \frac{\sum_{k=1}^{N} u_k(0)\, C_k}{\sum_{k=1}^{N} C_k} \qquad (i = 1 \cdots N). \tag{4.23}$$

From Theorem 4.2 we have

$$T^v_{Di} - T^v_{Dr} = \tau^v_{ir} \qquad (i = 1 \cdots N,\ i \neq r). \tag{4.24}$$

Further, because of conservation of charge, for each time t

$$\sum_{k=1}^{N} u_k(\infty)\, C_k = \sum_{k=1}^{N} u_k(t)\, C_k, \tag{4.25}$$

from which it follows that

$$\sum_{k=1}^{N} C_k \int_0^{\infty} [\, u_k(\infty) - u_k(t)\,]\, dt = 0 \tag{4.26}$$

or

$$\sum_{k=1}^{N} C_k T^v_{Dk} = 0. \tag{4.27}$$

From the set of N simultaneous equations that is formed by 4.24 and 4.27 it is possible to solve the values of T^v_{Di} $(i = 1 \cdots N)$. Similarly to the case where the RC network is connected to a voltage source, an appropriate transition time Tt_i for node i is then found from

$$Tt_i = \frac{2\, T^v_{Di}}{u_i(\infty) - u_i(0)}. \tag{4.28}$$

4.4.2.3 Computation of τ^v_{ij} for RC tree networks

For an RC tree network the constants τ^v_{ij} $(i = 1 \cdots N)$ that are necessary to compute the transition times for the voltage waveforms, may be computed in a time that is linear with the size of the network by using a two-pass algorithm [34, 27]. An implementation of such an algorithm is given by Algorithm 4.4.

Algorithm 4.4. Algorithm to compute τ_{ij}^{v} for all nodes i ($i = 1 \cdots N$) in an RC tree network.

```
# Node j is the root of the tree network
# For each node n, n.path denotes the parent of n,
# n.iv is the initial voltage of n, n.sv is the stable voltage of n,
# n.C is the capacitance of n, n.R is the resistance between n and its parent,
# and c is called a child of n if c.path = n.

  for all nodes n that are not a parent
    n.Q^L := 0
    repeat
      n.Q^L := n.Q^L + n.C * (n.sv - n.iv)
      n.path.Q^L := n.path.Q^L + n.Q^L
      n := n.path
    until one of the children c of n has not yet been visited or n.path = nil

  # now n is the root of the tree

  n.τ := 0
  for all children c of n
    assemble (c, n.τ)

procedure assemble (n, τ)

  n.τ := τ + n.r * n.C^L
  for all children c of n
    assemble (c, n)
```

The Algorithm 4.4 is based on the fact that for an RC tree network τ_{ij}^{v} may alternatively be written as

$$\tau_{ij}^{v} = \sum_{l \in \Gamma_{ij}} R_{p(l),l} \sum_{m \in \Omega_l} C_m \left(u_m(\infty) - u_m(0) \right), \tag{4.29}$$

where Γ_{ij} is the set of nodes that are on the path between i and j, including i and excluding j, $p(l)$ is the parent node of node l - i.e. the node that is connected to node l and that is closer the root node j -, and Ω_l is the set of nodes, including j that are in the sub-tree of node l - i.e. the nodes that are connected to node j via node l. From 4.29 it follows that

$$\tau_{ij}^{v} = \sum_{l \in \Gamma_{p(i),j}} R_{p(l),l} \sum_{m \in \Omega_l} C_m \left(u_m(\infty) - u_m(0) \right)$$

$$+ \; R_{p(i),i} \sum_{m \in \Omega_i} C_m \left(u_m(\infty) - u_m(0) \right)$$

$$= \tau_{ij}^{v} + R_{p(i),i} \, Q_i^L, \tag{4.30}$$

where

$$Q_i^L = \sum_{m \in \Omega_i} C_m \left(u_m(\infty) - u_m(0) \right) \tag{4.31}$$

is called the load charge of node i. During the first pass of Algorithm 4.4 the values for Q_i^L are computed by recurrently starting at a leaf node of the tree and then accumulating the contributions along the path towards j until a node is met of which all children have not yet been visited. During the second pass of Algorithm 4.4 the values τ_{ij}^{v} ($i = 1 \; \cdots \; N$) are incrementally assembled from the values at the parent nodes, the resistances at the nodes, and the load charges at the nodes, thereby starting at the root node j of the tree. Since each node is visited only once during each pass of the algorithm, the algorithm has a computation complexity that is linear with the number of nodes.

4.4.3 Connection with the switch-level model

In the following we will describe how the above theory is utilized in the switch-level model that was introduced in Section 4.2. Therefore we will consider the case where the relevant circuit part has transistors with an Undefined state, where it has Forced nodes with an X value, and/or where it has minimum and maximum node voltages that are not equal. We will present a solution that is based on an RC tree network model for the channel graph. Further - as a closer approximation of the reality - we will use in the formulas given in Section 4.4.2 for the stable voltages $u_i(\infty)$ of the nodes, the stable voltages that were found with the resistance division and charge-sharing algorithms (see Section 4.3). Also, at the end of this section, a few examples will be given that illustrate the accuracy of the timing model.

Suppose we want to find the rise times for the nodes of a channel graph that is connected to one or more Forced nodes that have a value I or X. This is achieved by first searching for each node in the channel graph for a best-case minimum resistance path to Forced with a logic value either I or X, thereby going through transistors that are Closed or Undefined, and a worst-case minimum resistance

path to Forced node with a logic value I, thereby going through transistors that are Closed. For the nodes that do not have a worst-case minimum resistance path to I an additional path search is done, thereby using the paths that were found from the above results and thereby also going through transistors that have an Undefined state. Thus, for both the worst-case path and the best-case path, the channel graph will automatically be partitioned as a tree network, whereby the root of one tree is respectively the coalescing of all Forced nodes with a value I, or the coalescing of all Forced nodes with either a value I or X.

Then the τ_{ij}^{v}'s are computed to find the rise times of the minimum and maximum voltage waveforms by applying the two-pass algorithm as given by Algorithm 4.4. In order to meet the requirements for the minimum and maximum voltage waveform, the maximum voltage waveforms are computed such that they rise as fast as possible, while the minimum voltage waveforms are computed such that they rise as slowly as possible. This is achieved as follows.

a. For the rise times of the maximum voltage waveforms, during the first pass of Algorithm 4.16, the charges $C_i^L = \sum_{m \in \Omega_i} C_m \, (sv_{m,\max} - iv_{m,\max})$ are accumulated based on the worst-case minimum resistance path. If the worst-case path goes through an Undefined state transistor, the accumulated charge is only carried to the next node if its value is less than zero. Then, during the second pass of Algorithm 4.16, the τ_{ij}^{v}'s are computed by backtracing the best-case minimum resistance path.

b. For the rise times of the minimum voltage waveforms during the first pass of Algorithm 4.16, the charges $C_i^L = \sum_{m \in \Omega_i} C_m \, (sv_{m,\min} - iv_{m,\min})$ are accumulated based on the best-case minimum resistance path. If the best-case path goes through an Undefined state transistor, the accumulated charge is only carried to the next node if its value is larger than zero. Then, during the second pass of Algorithm 4.16, the τ_{ij}^{v}'s are computed by backtracing the worst-case minimum resistance path.

In a similar way, the fall times are computed for the minimum and maximum voltage waveforms for the case when the channel graph is driven by at least one Forced node with a value O or X. The best-case and worst-case conditions are chosen to obtain slowly falling minimum voltage waveforms and to obtain fast falling maximum voltage waveforms.

When the channel graph does not have a Forced node with a value O or X connected to it when falling voltages are computed, or when it has not a Forced

node with a value I or X connected to it when rising voltages are computed, the transition times are evaluated based on the RC network model that is not connected to a voltage source. When computing the rise times, the node *k* with the largest value value $C_k\ iv_{\max}$ is thereby selected as the reference node, and when computing the fall times, the node *k* with the largest value $C_k\ (\ VDD - iv_{\min}\ )$ is thereby selected as the reference node. With the reference node being the root of the tree we then apply the same algorithms as for the case where the channel graph is connected to one or more Forced nodes.

To find an indication of the accuracy that can be obtained when using the switch-level timing model, the RC method has been applied to two different NMOS circuits to compute the delay times. The circuits are shown in Figure 4.19 and Figure 4.20. The resistances and capacitances of the transistor models that were used for these circuits were obtained from calibrations on the results that were obtained for inverter chains to which different kinds of network elements had subsequently been added [41]. For each node in a circuit, the delay time was measured as the time when the signal value reaches half the value of the supply voltage. In order to obtain an estimate of the accuracy of the results that were obtained when using the RC method, the results have been compared with the results that were obtained for the same circuit with the circuit simulator SPICE. The results are presented in Table 4.4, Table 4.5. Table 4.6 and Table 4.7. Note that the errors that are found in this case are about 10-20 %. In general, for circuits that differ markedly from standard MOS structures like and-or-invert gates and pass-transistor logic, a much larger error may be found.

4.4.4 Min-max delay simulation

Apart from modeling the X state of the nodes in the network, a different minimum and maximum voltage waveform for a node may also be used to model possible uncertainties in the transition times for the node. This can show the existence of circuit problems like race conditions and spikes. As an example we consider the subcircuit that is shown in Figure 4.21. If the designer has designed the total circuit such that at a certain moment first *a* O $\rightarrow$ I and then *b* O $\rightarrow$ I, then *c* I $\rightarrow$ O, the value of *d* will remain I, and the final value of *c* will be O. If - because of a slight difference in delay times in the rest of the circuit - first *b* O $\rightarrow$ I and then *a* O $\rightarrow$ I, then *d* will hold the value of *c* at I, and the final value of *c* will be I. Hence a critical race condition exists for the signals *a* and *b* which causes that the value of *c* is dependent on the relative delay times in the circuit.

Generally, a problem as described above will not be revealed by means of simulation since the relative delay times in the circuit may turn out as required for

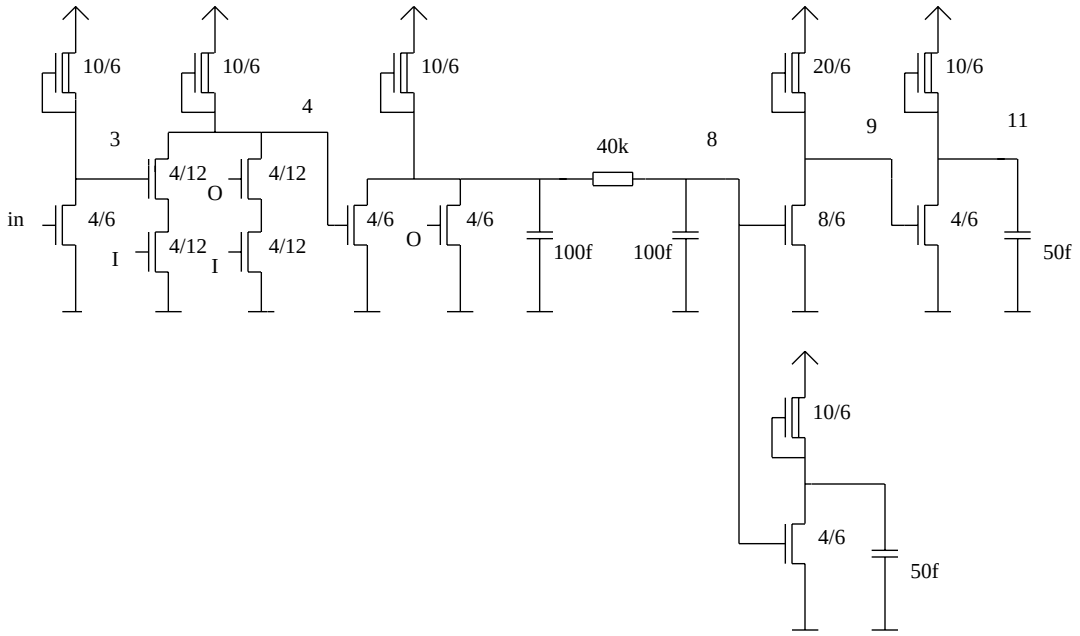


Figure 4.19. Test circuit 1.

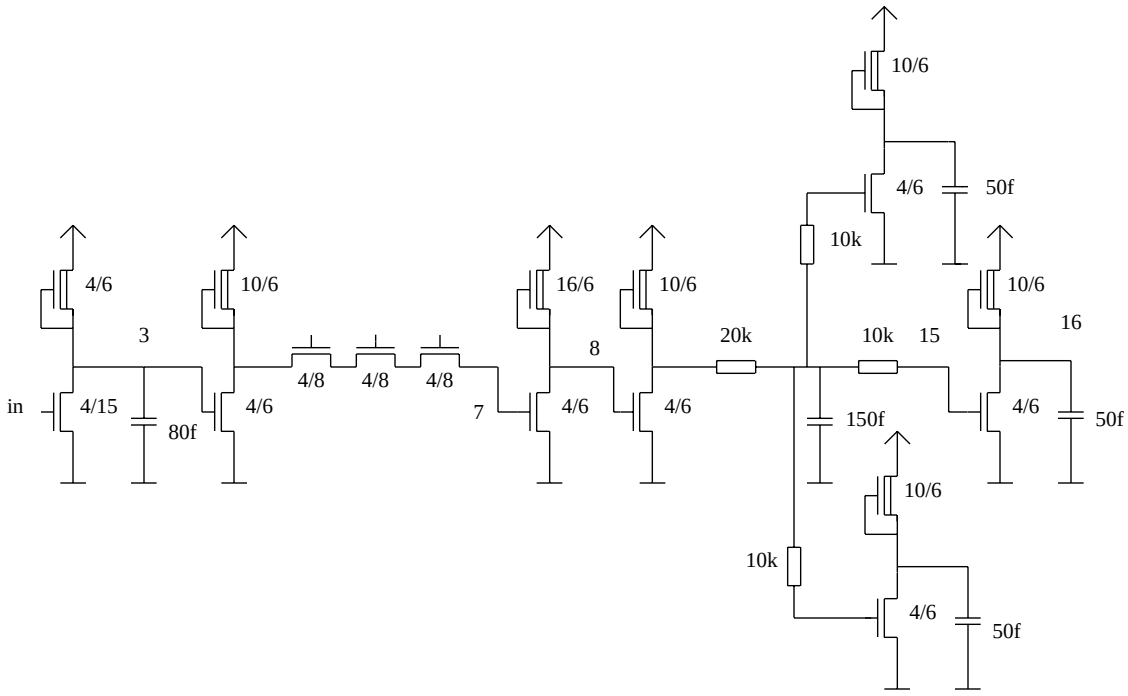


Figure 4.20. Test circuit 2.

the correct behavior of the circuit, see the simulation results for the latch circuit that are shown in Figure 4.22. One way of detecting the above type of possible

Table 4.4. Delay times for the circuit in Figure 4.19, rising input.

	delay in nsec.		
node	RC-method	SPICE	% error
3	0.8	0.6	25.0
4	2.2	2.1	4.8
8	10.1	8.8	14.7
9	12.7	11.2	13.4
11	13.9	12.9	7.7

Table 4.5. Delay times for the circuit in Figure 4.19, falling input.

	delay in nsec.		
node	RC-method	SPICE	% error
3	1.5	1.6	-6.3
4	2.1	2.0	5.0
8	19.8	16.1	23.0
9	21.9	21.2	3.3
11	25.1	23.7	5.9

Table 4.6. Delay times for the circuit in Figure 4.20, rising input.

	delay in nsec.		
node	RC-method	SPICE	% error
3	0.6	0.6	0.0
7	4.7	5.1	-7.8
8	5.4	5.8	-6.9
15	18.8	17.2	9.3
16	20.0	19.7	1.5

Table 4.7. Delay times for the circuit in Figure 4.20, falling input.

	delay in nsec.		
node	RC-method	SPICE	% error
3	2.2	2.4	-8.3
7	4.1	3.9	5.1
8	6.0	5.6	7.1
15	12.0	10.2	17.6
16	15.2	13.7	10.9

design errors is to use a ternary simulation model [42, 43]. In this model, when at a particular moment one or more inputs changes its value, the values of these

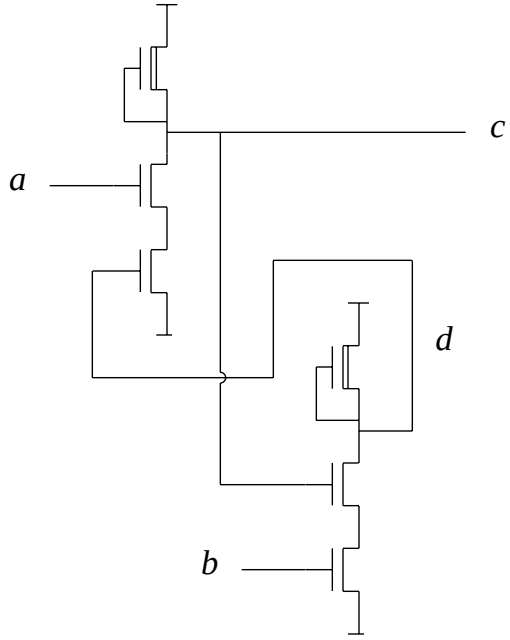


Figure 4.21. A latch circuit with race conditions.

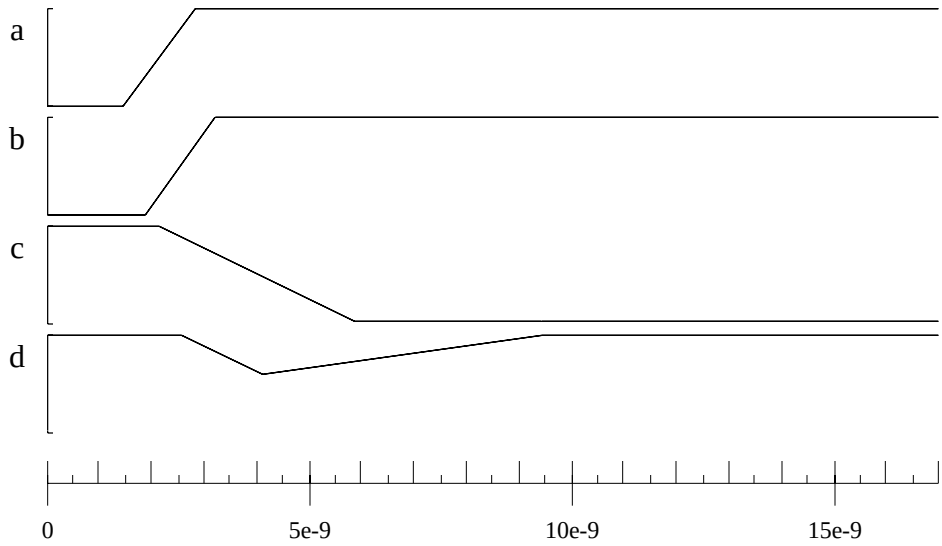


Figure 4.22. A correct simulation result for the circuit in Figure 4.21.

inputs are first set to X. During this phase, which is called the transition phase, a stabilization of the network is computed in which the X value is assigned to all nodes that are dependent on the values of the inputs that are changing. Then, during a second phase, which is called the stabilization phase, the final values are applied to the inputs. Now, all nodes in the network that became X during the transition phase become O or I, except for the nodes whose final value is not

explicitedly determined by the value(s) of one or more input nodes. Thus, for all nodes in the network whose final value is dependent on the relative propagation times in the circuit, the final value will be X. These will be the nodes that are in a feedback loop in which an X value has been locked during the transition phase (see Figure 4.23a), the nodes that have become X during the transition phase and that become isolated from Forced nodes during the stabilization phase (see Figure 4.23b), and all nodes whose value is dependent on the value of these nodes. As an example Table 4.8 shows the signal values that are subsequently computed for the latch circuit when applying a ternary simulation model.

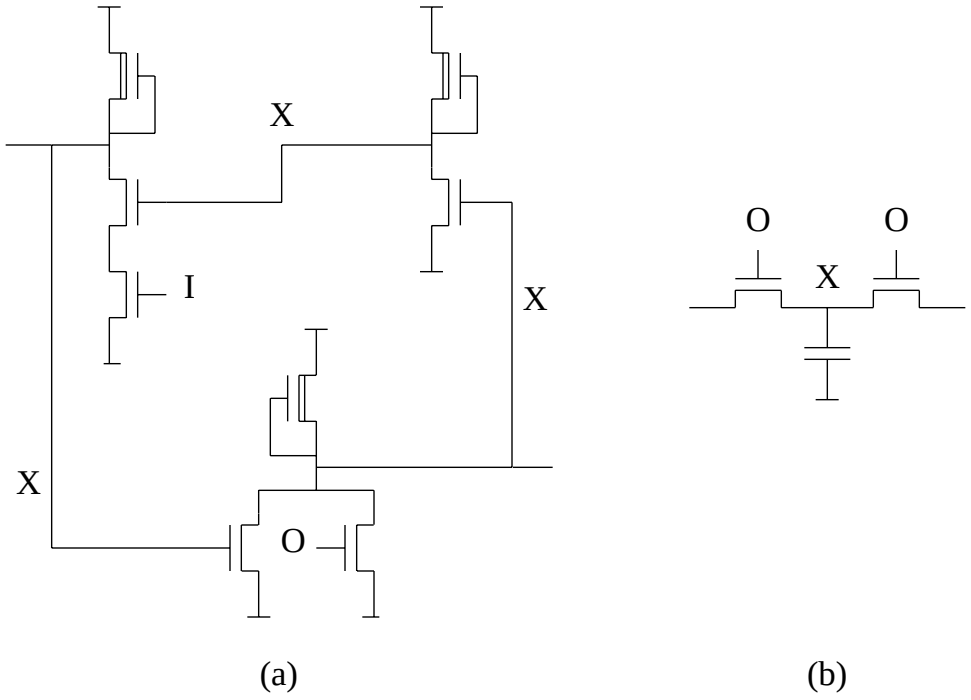


Figure 4.23. Different possibilities to store an X value in a circuit, (a) in a feedback loop, (b) isolated from inputs nodes.

With the ternary simulation model it is possible to detect every race condition that occurs for the specified input pattern. Moreover, a potential spike on a node is indicated by a sequence of the form O-X-O or I-X-I. A disadvantage of the ternary simulation model however is that does not take into account the qualitative aspects of the race conditions. For circuits that have race conditions that, in practice, will not give any problems - because of practical upper and lower bounds for the delay times - the method will produce simulation results that may be much too pessimistic. A more realistic model is therefore obtained by using a delay model in which minimum and maximum values are employed for the delay times.

Table 4.8. Signal values that are computed for the latch circuit shown in Figure 4.21 when using a ternary simulation model.

	a	b	c	d
initial	O	O	I	I
transition phase	X	X	I	I
	X	X	X	X
stabilization phase	I	I	X	X

This may be achieved in the switch-level timing model by using different transition times for the minimum and maximum voltage waveforms. The rising maximum voltage waveforms and the falling minimum voltage waveforms are thereby multiplied by a minimum deviation factor to speed them up by a proportional value, and the falling maximum voltage waveforms and the rising minimum voltage waveforms are thereby multiplied by a maximum deviation factor to slow them down by a proportional value. Then, for each signal transition, the signal will have an intermediate X value, and the X value will be the final value if (1) the node is in a feedback loop in the circuit in which X value is present long enough to be locked in the loop, (2) the node becomes isolated from a Forced node while it is an X state, or (3) the value of the node is dependent on the value of one or more nodes that are mentioned under (1) and (2). Also, an X value will appear as a temporary value at a node that would have normally remained O or I, indicating a potential spike, if one or more signals that control the value of the node are X for a sufficiently long period of time.

The above is illustrated by the simulation results for the latch circuit that are shown in Figure 4.24. For the simulation results shown in Figure 4.24, the same input pattern was applied to the latch circuit as that shown in Figure 4.22, but now the transition times that are found during simulation have been multiplied by 0.85 to obtain a minimum value and by 1.15 to obtain a maximum value. From the results shown in Figure 4.24 it is noted that at a certain moment both node *c* and node *d* have an X value, and that the X value is locked in the loop *c*-*d*. In Figure 4.25 the same simulation results are presented, but now for the case where the difference in the delay between node *a* and and node *b* has been increased. For this case no particular moment exists during which both *c* and *d* are X, and the circuit will reach a final state where *c* and *d* have the correct values. Thus in this case the race condition not critical. However, at node *d* a spike is visible that might cause some problems in other parts of the circuit. The limitations of the switch-level min-max delay model are that (1) the model does not take into account a possible correlation between the different signals that cause the race

condition, and (2) the algorithms that update the spline pairs are based on best-case/worst-case conditions that on some occasions may produce simulation results that are overly pessimistic.

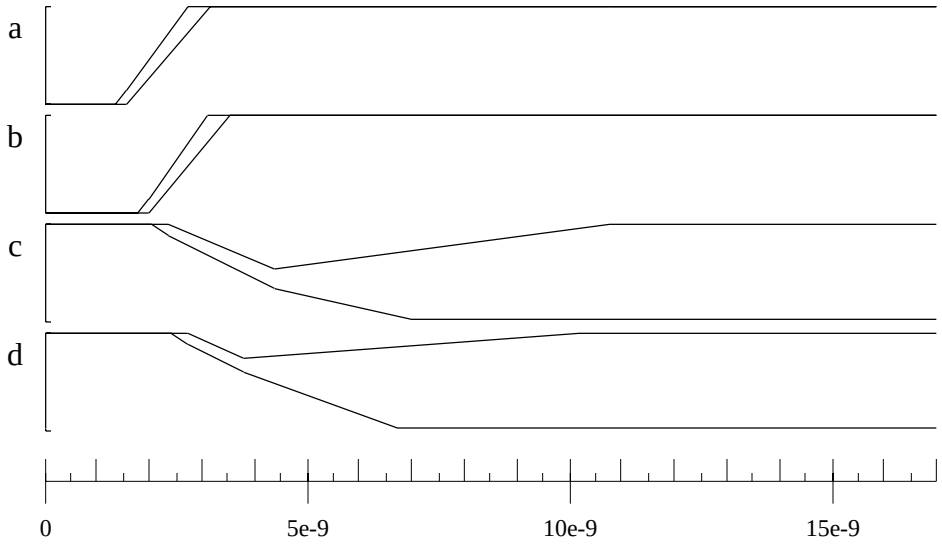


Figure 4.24. A min-max delay simulation that yields X values because of a critical race condition.

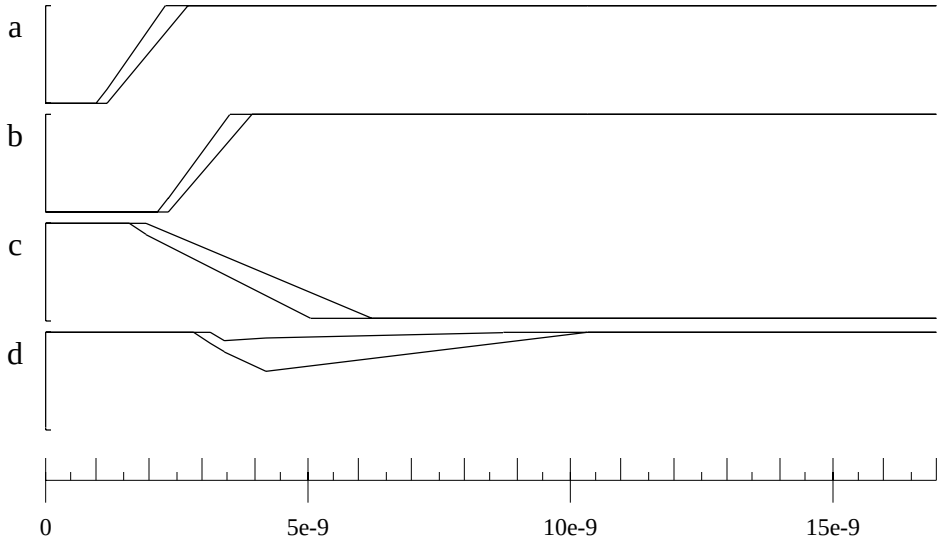


Figure 4.25. A min-max delay simulation that produces correct simulation results.

4.5 Functional simulation

With functional simulation, the circuit that is simulated is divided up into parts that have a specific (high-level) function - e.g. multipliers, full-adders and registers - and the behavior of each of these parts is described in a high-level description language. In addition, a gate-level simulation may be considered as a special form of functional simulation where the circuit is divided up into small subcircuits like NANDs and NORs, and where the behavior of each of these subcircuits has been predefined. A functional simulation, or a mixed transistor, gate and/or functional level simulation, is executed (1) to simulate circuits that are not yet (fully) implemented at the transistor level or (2), if the circuit parts are, without significant loss of accuracy, replaced by their functional equivalence, to increase the simulation speed. Further, since the circuit is described at different levels of abstraction, a mixed functional, gate and transistor level simulator is a valuable verification tool during many stages of the design process.

Below we will describe how the switch-level model that was introduced in the previous sections is extended to support functional simulation, and thus also gate-level simulation. In Section 4.5.1 the behavioral model is described that is used to model the behavior of parts of digital circuits, and in Section 4.5.2 how this behavioral model is connected with the switch-level network model, e.g. to model output resistance and delay times, is discussed.

4.5.1 Behavioral model

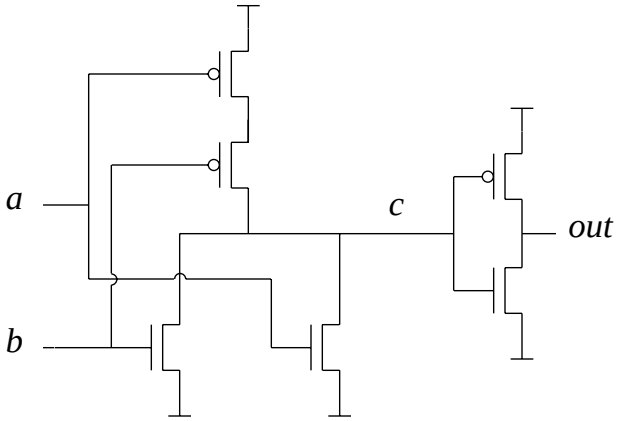
A part of a circuit that is described at the functional level can be considered as a network element that, according to some high-level behavioral description, defines a relation between the values of its input terminals, the values of its state variables and the values of its output terminals [44, 45]. When I denotes the vector of the values of the input terminals, O denotes the vector of the values of the output terminals, and S is the vector of the values of the state variables of the function block, then the function block defines the values of the output terminals and the state variables to be dependent on the values of the input terminals and the state variables according to

$$[\, O \; S \,]^T = f\,(I,\; S). \tag{4.32}$$

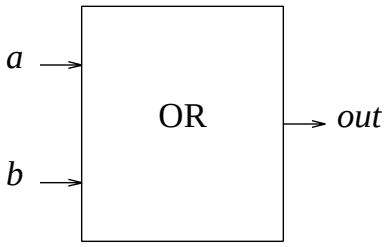
The function f is in practice defined in a high-level description language like C or Pascal [10, 46]. When the function block describes an asynchronous cell, expression 4.32 is evaluated each time when the value of one the input terminals or one of the state variables changes. When the function block describes a clocked or synchronous cell (i.e. a finite state machine [26]) expression 4.32 is evaluated

only when the value of an input terminal that is a clock signal changes. Further, with each value transition of a state variable or an output terminal, a delay time may be associated. As an example, Figure 4.26b shows the behavioral model for the OR circuit shown in Figure 4.26a, where the function *f* is described as follows:

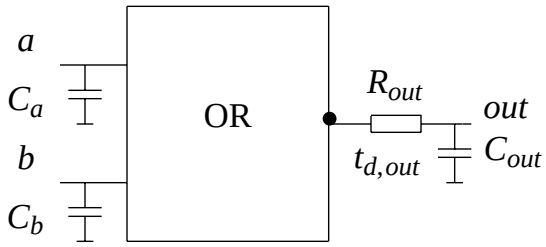
$$out = a \textbf{ or } b. \tag{4.33}$$



(a)



(b)



(c)

Figure 4.26. CMOS OR cell (a), its functional representation (b), and the network model of the function block (c).

4.5.2 Network model

Each instance of a function block stores a set of values that is given by the values of the state variables of the function block and the values of the output terminals of the function block. In the network, each of these values may be modeled by a node that has a value equal to the value that is stored by the state variable or output terminal. The nodes are of type Forced since their value is unilaterally determined from the values of the input terminals and the state variables of the

function block. The function block interacts with the other parts of the network via its input terminals and via its output terminals. To model the output resistance of each output terminal, a resistor may be associated with each output of a function block, and to model the capacitive load of the input terminals and the output terminals, a capacitor may be associated with each input terminal and each output terminal of a function block. Based upon the above notations, the function block is represented in the switch-level simulation model by a network element shown in Figure 4.27. For the network model of the function block shown in Figure 4.27, which has K inputs, L state variables and M outputs, $i1$, $i2$, $\cdots$ iK are the input terminals of the function block, $s1$, $s2$, $\cdots$ sL are the nodes that store the values of the state variables of the function block, $v1$, $v2$, $\cdots$ vM are the nodes that store the values of the output terminals of the function block, $o1$, $o2$, $\cdots$ oM are the output terminals themselves, C_{i1}, C_{i2}, $\cdots$ C_{iK} represent the load capacitances of the function block at the input terminals, C_{o1}, C_{o2}, $\cdots$ C_{oM} represent the load capacitances of the function block at the output terminals, R_{o1}, R_{o2}, $\cdots$ R_{oM} are the output resistances of the output terminals, $t_{d,s1}$, $t_{d,s2}$, $\cdots$ $t_{d,sK}$ are the delay times that associated with the value transitions at the state variables, and $t_{d,o1}$, $t_{d,o2}$, $\cdots$ $t_{d,on}$ are the delay times that are associated with the value transitions at the output terminals.

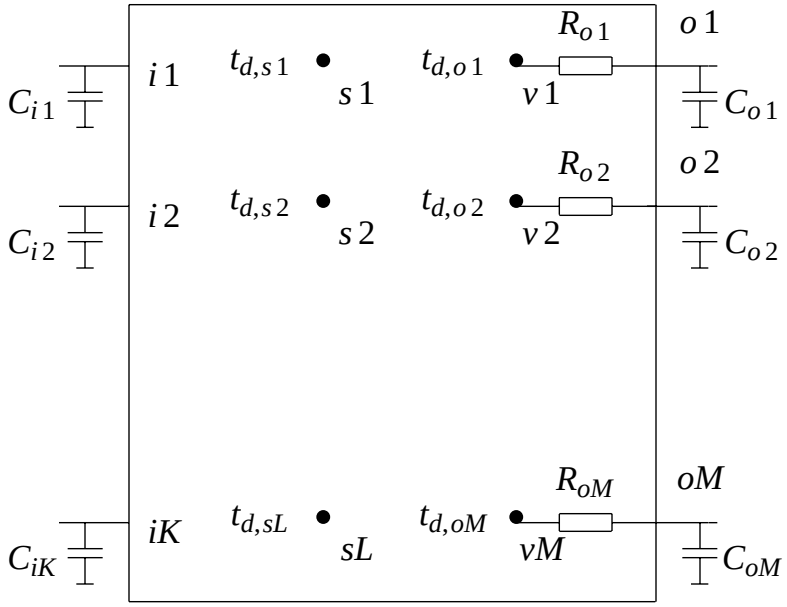


Figure 4.27. General function block network model.

The values of each of the parameters shown in Figure 4.27 may be dependent on the values of the input terminals and the values of the state variables. To accurately model delay times for situations where multiple events occur for the

inputs of one function block within a short period of time, a similar state description may be used for the internal nodes of the function block as for the other nodes in the network (see Section 4.2.4).

As an example of a network model for a function block, Figure 4.26c shows the network model for the function block that is shown in Figure 4.26b. The capacitance C_a represents the gate capacitance of the upper p-enhancement transistor of the OR gate in Figure 4.26a plus the gate capacitance of the right n-enhancement transistor of the OR gate in Figure 4.26a. The capacitance C_b represents the gate capacitance of the lower p-enhancement transistor of the OR gate in Figure 4.26a plus the gate capacitance of the left n-enhancement transistor of the OR gate in Figure 4.26a. The capacitance C_{out} represents the sum of the drain-source capacitances at node *out* in Figure 4.26a. The resistance R_{out} represents either the impedance of the p-enhancement transistor of the inverter in Figure 4.26a or the impedance of the n-enhancement transistor of the inverter in Figure 4.26a, depending on the value of the output *out*. The delay time $t_{d,out}$ in Figure 4.26c is equal to the delay time between node *a* (or node *b*) and node *c* in Figure 4.26a.

Algorithm 4.5 shows the extension for the switch-level simulation mechanism that is necessary to simulate networks that also contain function blocks. Algorithm 4.5 consists of the simulation step given by Algorithm 4.1, but now with an extension to evaluate the different instances of the function blocks that occur in the network. With Algorithm 4.5 it is assumed that the state variables and the output values of each instance of a function block are implemented as nodes. In order to prevent the result of an evaluation of one function block influencing the result of an evaluation of another function block within the same simulation step, a variable 'value' is introduced for each node that stores the current value for that node. The variable 'value' of a node is updated at the beginning of a new simulation step when an event occurs for that node.

4.6 A practical switch-level simulation program

Most of the above simulation models and algorithms have been implemented in a switch-level simulation program called SLS [27, 24, 47, 48, 49]. The SLS simulator has been integrated in the Nelsis VLSI design system [50, 51] and it has been used extensively during the design of many (large) digital MOS circuits [52].

The usage of the SLS simulator within the design system is illustrated in Figure 4.28. The circuit that is simulated by SLS is described in the data base by a hierarchical network description containing transistors, resistors, capacitors, gate-level elements and/or function blocks. Information about this description is

Algorithm 4.5. Modified procedure simulation_step to simulate networks with function blocks.

---

procedure simulation_step ( )

S will contain a set of nodes that is such that of each channel graph
that needs to be evaluated at least one node is present in *S*.
It is assumed that initially *n.evalflag* = 0 for all nodes *n*
F will contain the set of function instances that should be evaluated.

$S := \emptyset$, $F := \emptyset$

while time_next_event ( ) = t_c
  δ := get_next_event ( )
  δ.*n.value* := δ.*val*
  for all transistors *t* with *t.gate* = δ.*n*
    update *t.state* according to δ.*val*
    if *t.drain.type* = *Normal* $S := S \cup$ *t.drain*
    if *t.source.type* = *Normal* $S := S \cup$ *t.source*
  if δ.*n.type* = *Forced*
    for all transistors *t* with *t.drain* = δ.*n* **and** *t.state* $\neq$ *Open*
      if *t.source.type* = *Normal* $S := S \cup$ *t.source*
    for all transistors *t* with *t.source* = δ.*n* **and** *t.state* $\neq$ *Open*
      if *t.drain.type* = *Normal* $S := S \cup$ *t.drain*
  for all function instances *f* that have an input connected to *n*
    or that have a state variable that is *n*
    $F := F \cup f$

for all nodes *n* in *S*
  n.evalflag := 1

for all nodes *n* in *S*
  if *n.evalflag* = 1
    search channel graph *g* of node *n*
    evaluate channel graph *g*
    for all nodes *m* in channel graph *g*
      m.evalflag := 0

for all function instances *f* in *F*
  evaluate function instance *f*

---

maintained by the design system such as the different parts in which the total circuit is subdivided, their creation dates, how these parts were created, and their verification statuses [51]. One possibility to enter (a part of) a circuit description into the data base is by providing a textual description of the circuit. This option is used to describe the behavior of function blocks and to the transport circuit descriptions between different design data bases. Another possibility consists of drawing the circuit diagram of the circuit with a schematic entry program [53]. A third possibility is by using a layout-to-circuit extraction program [54], which is applied to obtain the circuit from its layout description, including layout parasitics such as wire capacitances and wire resistances.

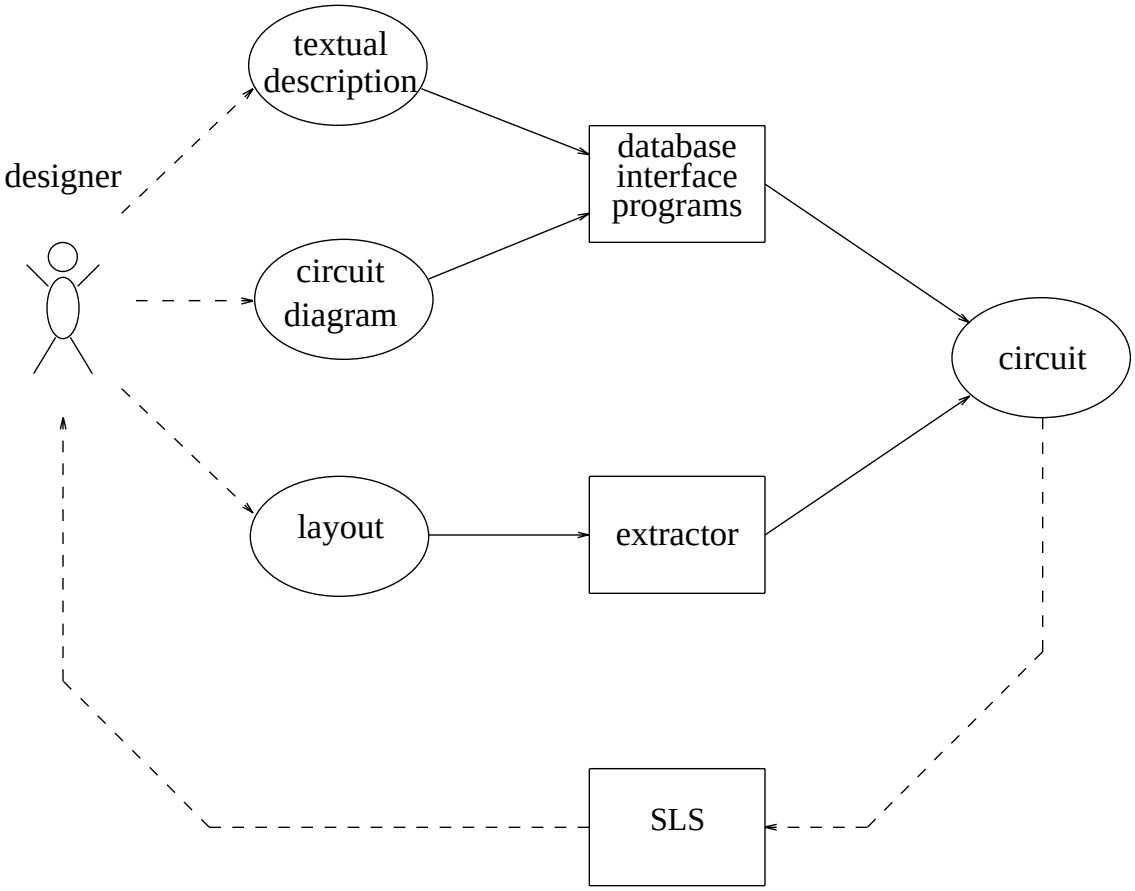


Figure 4.28. The program SLS in a design environment.

To simulate a circuit, the SLS simulator can be used at three different levels. The first level uses a simulation model as described in [18] and models transistor impedances and node capacitances by abstract values. This level is used to simulate circuits for which no transistor dimensions and network capacitances are available or for which they are irrelevant. The second level uses the resistance

division and charge-sharing algorithms described in Section 4.3, and it takes into account the precise parameters for the transistor impedances and the node capacitances. This level is used to simulate circuits for which correct behavior is dependent on resistance division effects between transistors of the same type and/or on charge-sharing effects. The third level extends the second level in that it computes also the rise and fall times of the signals. This level is used to simulate the logic behavior as well as the timing behavior of the circuit. Additionally, a min-max delay simulation can be done at level 3 to study the influence of parameter variations on the simulation results and to verify the circuit on race conditions. This requires the specification of a minimum proportional value for the signal transition times, and a specification of a maximum proportional value for the signal transition times.

Figure 4.29 shows simulation output for an 8-bit NMOS random counter circuit, containing 149 transistors, that was simulated using SPICE version 2g3. The simulation took 39 minutes and 41.3 seconds CPU time on an HP-9000/840 computer. As a comparison, Figure 4.30 shows the results for the same simulation, but now for the case where the circuit was simulated using SLS at level 3. In this case, the simulation took, also on an HP-9000/840 computer, only 2.4 seconds CPU time. In Table 4.9 CPU times are shown for some other typical simulations carried out with SLS. All times were obtained on an HP-9000/840 computer. Filter, multiplier and cordic [55] were all fully simulated at the transistor level, while clp [52], in addition, had its environment simulated at the functional level.

Table 4.9. CPU times for simulations using SLS on an HP-9000/840 computer.

circuit	transistors	clock cycles	level	CPU time (min:sec)
filter	1278	50	1	7.6
	1278	50	2	14.9
	1278	50	3	22.7
multiplier	5376	22	3	1:05
cordic processor	63416	75	3	64:09
clp processor	20082	2442	2	386:36

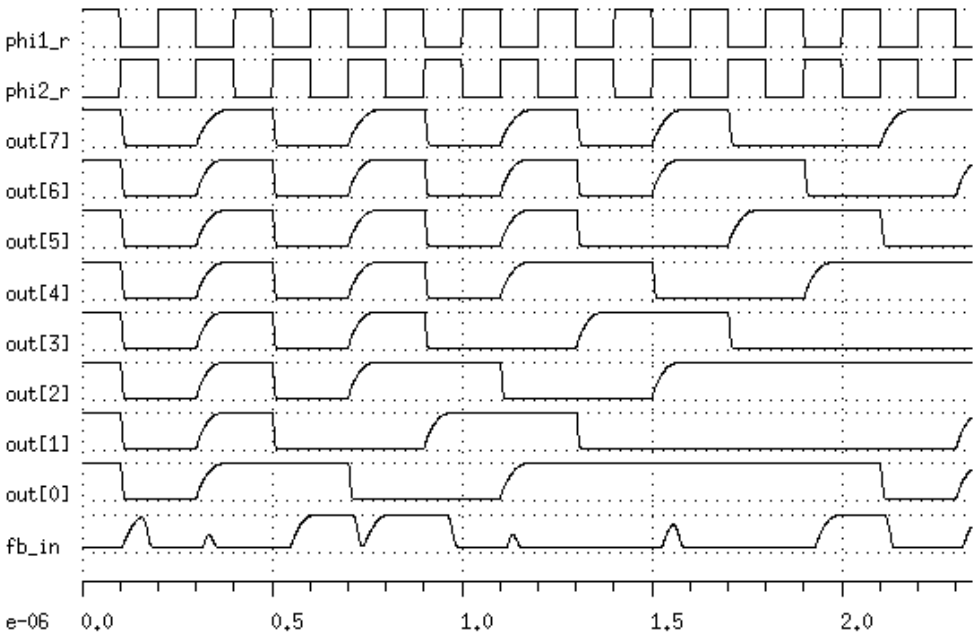


Figure 4.29. Simulation results for an 8-bit random counter circuit using SPICE (39 min. 41.3 sec. CPU time on an HP-9000/840 computer).

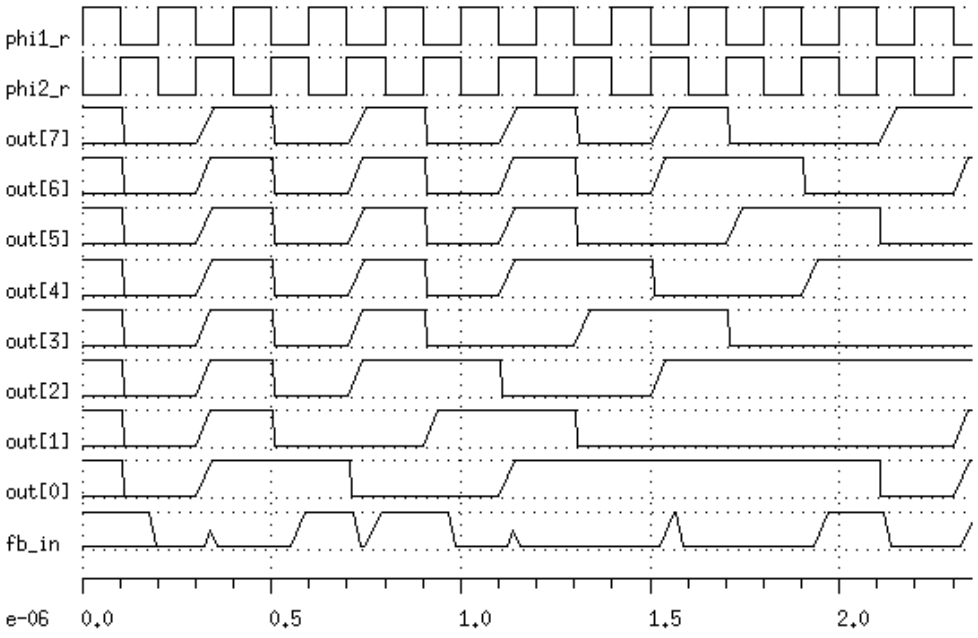


Figure 4.30. Simulation results for an 8-bit random counter circuit using SLS (2.4 sec. CPU time on an HP-9000/840 computer).

References

1. L.W. Nagel, ‘‘SPICE2: A computer program to simulate semi-conductor circuits,’’ ERL Memo. ERL-M520, Electronics Res. Lab., Univ. California, Berkeley (1975).

2. W.T. Weeks, A.J. Jiminez, G.W. Mahoney, D. Mehta, H. Quassemzadeh, and T.R. Scott, ‘‘Algorithms for ASTAP - A Network Analysis Program,’’ *IEEE Trans. Circuit Theory* **CT-20** pp. 628-634 (Nov. 1973).

3. B.R. Chawla, H.K. Gummel, and P. Kozak, ‘‘MOTIS - an MOS timing simulator,’’ *IEEE Trans. on Circuits and Systems* **CAS-22**(12) pp. 901-910 (Dec. 1975).

4. G. Arnout and H.J. de Man, ‘‘The Use of Threshold Functions and Boolean Controlled Network Elements for Macromodelling of LSI Circuits,’’ *IEEE Journal Solid-State Circuits* **SC-13** pp. 326-332 (June 1978).

5. A.R. Newton, ‘‘Techniques for the Simulation of Large-Scale Integrated Circuits,’’ *IEEE Trans. on Circuits and Systems* **CAS-26** pp. 741-749 (Sept. 1979).

6. C.F. Chen, C-Y Lo, H.N. Nham, and Prasad Subramaniam, ‘‘The second generation MOTIS mixed-mode simulator,’’ *Proc. 21st Design Automation Conference*, Albuquerque, New Mexico, pp. 10-16 (June 1984).

7. B.H. Scheff and S.P. Young, ‘‘Gate-Level Logic Simulation,’’ in *Design Automation of Digital Systems*, ed. M.A. Breuer, Prentice-Hall, Englewood Cliffs (1972).

8. S.A. Szygenda and E.W. Thompson, ‘‘Modeling and Digital Simulation for Design Verification and Diagnosis,’’ *IEEE Trans. on Computers*, pp. 1242-1253 (Dec. 1976).

9. W.M. vanCleemput, ‘‘Computer Hardware Description Languages and their Applications,’’ *Proc. 16th DAC*, pp. 554-560 (June 1979).

10. E.J. Frey, ‘‘ESIM: A Functional Level Simulation Tool,’’ *Proc. ICCAD-84*, pp. 48-50 (Nov. 1984).

11. E. Lelarasmee, A.E. Ruehli, and A.L. Sangiovanni-Vincentelli, ‘‘The waveform relaxation method for time-domain analysis of large scale integrated circuits and systems,’’ *IEEE Trans. on CAD* **CAD-1**(3) pp. 131-145 (July 1982).

12. W.M.G. van Bokhoven, ‘‘Piecewise Linear Analysis and Simulation,’’ pp. 129-166 in *Circuit Analysis, Simulation and Design, part 2*, ed. A.E. Ruehli, North-Holland, Amterdam, the Netherlands (1987).

13. R.E. Bryant, ‘‘An algorithm for MOS logic simulation,’’ *Lambda Magazine*, pp. 46-53 (4th Quarter 1980).

14. C.J. Terman, ‘‘RSIM - A logic-level timing simulator,’’ *Proc. IEEE ICCD-83*, New York, pp. 437-440 (Oct. 1983).

15. R.E. Bryant, ‘‘A Survey of Switch-Level Algorithms,’’ *IEEE Design & Test*, (August 1987).

16. J. Rubinstein, P. Penfield, and M.A. Horowitz, ‘‘Signal delay in RC tree networks,’’ *IEEE Trans. on CAD* **CAD-2**(3) pp. 202-211 (July 1983).

17. V. Ramachandran, ‘‘An Improved Switch-Level Simulator for MOS Circuits,’’ *Proc. Design Automation Conference*, pp. 293-299 (1983).

18. R.E. Bryant, ‘‘A switch-level model and simulator for MOS digital systems,’’ *IEEE Trans. on Computers* **C-33**(2) pp. 160-177 (Feb. 1984).

19. Z. Barzilai, D.K. Beece, L.M. Huisman, V.S. Iyengar, and G.M.Silberman, ‘‘SLS - A Fast Switch-Level Simulator,’’ *IEEE Trans. on CAD* **CAD-7**(8) pp. 838-849 (August 1988).

20. R.E. Bryant, D. Beatty, and K. Brace, ‘‘COSMOS: A Compiled Simulator for MOS Circuits,’’ *Proc. 24th Design Automation Conference*, pp. 9-16 (1987).

21. T.J. Schaefer, ‘‘A Transistor-Level Logic-with-Timing Simulator for MOS circuits,’’ *Proc. 22nd Design Automation Conference*, pp. 762-765 (1985).

22. P.M. Dewilde, A.J. van Genderen, and A.C. de Graaf, ‘‘Switch level timing simulation,’’ *Proc. IEEE ICCAD-85*, Santa Clara, (Nov. 1985).

23. A. Salz and M. Horowitz, ‘‘IRSIM: An Incremental MOS Switch-Level Simulator,’’ *Proc. 26th Design Automation Conference*, pp. 173-178 (1989).

24. A.J. van Genderen, ‘‘SLS: An Efficient Switch-Level Timing Simulator Using Min-Max Voltage Waveforms,’’ *Proc. VLSI 89 Conference*, Munich, FRG, pp. 79-88 (August 1989).

25. D. Adler, ‘‘Switch-Level Simulation Using Dynamic Graph Algorithms,’’ *IEEE Trans. on Computer-Aided Design* **CAD-10**(3) (March 1991).

26. C.A. Mead and L.A. Conway, *Introduction to VLSI systems,* Addison Wesley, Reading, Mass. (1980).

27. A.J. van Genderen and A.C. de Graaf, ‘‘SLS: A Switch-Level Timing Simulator,’’ pp. 2.93-2.146 in *The Integrated Circuit Design Book,* ed. P. Dewilde, Delft University Press, Delft, the Netherlands (1986).

28. H. Cai, ‘‘Routing Channels in VLSI Layout,’’ *Ph.D. Thesis*, Delft, the Netherlands, (March 1989).

29. N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: A System Perspective,* Addison-Wesley, Reading (1985).

30. C. Svensson and R. Tjarnstrom, ‘‘Switch-Level Simulation and the Pass Transistor Exor Gate,’’ *IEEE Trans. on Computer-Aided Design* **CAD-7**(9) pp. 994-997 (Sep. 1988).

31. L. Spaanenburg, *Personal Communications.*

32. P. Penfield, Jr. and J. Rubinstein, ‘‘Signal Delay in RC Tree Networks,’’ *Proc. 18th Design Automation Conference*, Nashville, pp. 613-617 (June/July 1981).

33. R. Putatunda, ‘‘Autodelay: A Second-Generation Automatic Delay Calculation Program for LSI/VLSI Chips,’’ *Proc. ICCAD-84,* pp. 188-190 (Nov. 1984).

34. T.M. Lin and C.A. Mead, ‘‘Signal Delay in General RC Networks,’’ *IEEE Trans. on Computer Aided Design* **CAD-3**(4) pp. 331-349 (Oct. 1984).

35. J.L. Wyatt, ‘‘Signal Delay in RC Mesh Networks,’’ *IEEE Trans. on Circuits and Systems* **CAS-32**(5) pp. 507-510 (May 1985).

36. J.K. Ousterhout, ‘‘A switch-level timing verifier for digital MOS VLSI,’’ *IEEE Trans. on CAD* **CAD-4**(3) pp. 336-349 (July 1985).

37. N.P. Jouppi, ‘‘Timing Analysis and Performance Improvement of MOS VLSI Design,’’ *Trans. on Computer-Aided Design* **CAD-6**(4) pp. 650-665 (July 1987).

38. P.K. Chan and K. Karplus, ‘‘Computing Signal Delay in General RC Networks by Tree/Link Partitioning,’’ *Proc. 26th Design Automation Conference,* Las Vegas, Nevada, pp. 485-490 (1989).

39. C.-Y. Chu and M.A. Horowitz, ‘‘Charge-Sharing Models for Switch-Level Simulation,’’ *IEEE Trans. on Computer-Aided Design* **CAD-6**(6) pp. 1053-

1061 (Nov. 1987).

40. W.C. Elmore, ‘‘The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers,’’ *J. Applied Physics* **19** pp. 55-63 (Jan. 1948).

41. A.J. Schooneveld, ‘‘Determination of SLS Transistor Model Parameters,’’ Technical Report 87-84, Delft University of Technology, Network Theory Section, Delft, the Netherlands (May 1987).

42. E.B. Eichelberger, ‘‘Hazard Detection in Combinational and Sequential Switching Circuits,’’ *IBM Journal of Research and Development,* pp. 90-99 (March, 1965).

43. R.E. Bryant, ‘‘Race Detection in MOS Circuits By Ternary Simulation,’’ *VLSI 83*, pp. 85-95 (1983).

44. D.R. Coelho, *The VHDL Handbook,* Kluwer Academic Publishers, Dordrecht, the Netherlands (1989).

45. E. Sternheim, R. Singh, and Y. Trivedi, *Digital Design with Verilog HDL,* Automata Publishing Company, Cupertino, California (1990).

46. O. Hol, ‘‘Mixed Level Simulation: Using Function Blocks in SLS,’’ Report 88-81, Delft University of Technology, Network Theory Section, Delft, the Netherlands (Feb. 1988).

47. A.J. van Genderen, ‘‘SLS: A Switch-Level Timing Simulator,’’ *Delft Progr. Rep.* **12** Delft, pp. 280-290 (1988).

48. A.C. de Graaf and A.J. van Genderen, *SLS: Switch-Level Simulator User’s Manual,* Delft University of Technology, Network Theory Section, Delft, the Netherlands (1985,1988).

49. O. Hol, *FUNC_MKDB Function Blocks in SLS, User’s Manual*, Delft University of Technology, Network Theory Section, Delft, the Netherlands (Feb. 1988).

50. P. Dewilde, ed., *The integrated circuit design book: Papers on VLSI design methodology from the ICD-NELSIS Project,* Delft University Press, Delft, the Netherlands (1986).

51. P. van der Wolf, P. Bingley, and P. Dewilde, ‘‘On the Architecture of a CAD Framework: The NELSIS Approach,’’ *Proc. IEEE 1st European Design Automation Conference,* (1990).

52. O.E. Herrmann and B.J.F. van Beijnum (editors), *Lecture Notes of the Nelsis-project*, Technical University Twente (March 9-11, 1988).

53. A. Lodder, M. van Stiphout, and J.T.J. van Eijndhoven, ‘‘The Eindhoven Schematic Editor,’’ pp. 1.61-1.68 in *The Integrated Circuit Design Book*, ed. P. Dewilde, Delft University Press, Delft, the Netherlands (1986).

54. N.P. van der Meijs and A.J. van Genderen, ‘‘Space: An Accurate and Efficient Extractor for Submicron Integrated Circuits,’’ *Delft Progr. Rep.* **12** Delft, pp. 260-279 (1988).

55. A.A.J. de Lange, A.J. van der Hoeven, E.F. Deprettere, and J. Bu, ‘‘An Optimal Floating-Point Pipeline CMOS CORDIC Processor: Algorithm, Automated Design, Layout and Performance,’’ *Proc. ISCAS-88*, Espoo, Finland, pp. 2043-2047 (June 7-9, 1988).

Summary

Because of the great complexity and the high fabrication costs of VLSI circuits, it is necessary to thoroughly verify the behavior of these circuits before they are fabricated. In addition to obvious errors such as wrong or incomplete connections between different parts of the circuit, electrical errors may occur because of physical limitations such as too great a drop in voltage along supply lines, too large delay time values, and unintended capacitive coupling effects between different interconnections. In order to efficiently and accurately verify a design of an integrated circuit for these types of errors, low-complexity but sufficiently accurate models are required that are easily incorporated in computer programs for Computer Aided Design (CAD). In this thesis, three such modeling techniques are described each of which models some important aspect of the behavior of (V)LSI circuits.

In Chapter 2 the modeling of the resistive effects of the interconnections in integrated circuits is first discussed. In addition to the resistances of the interconnections, also the distributed capacitive effects are important in characterizing the transmission behavior of the interconnections, and are covered next. In this thesis, a modeling technique is described that is based on a finite-element technique and that allows one to find low-complexity, lumped, RC models that accurately represent the resistive effects of the interconnections and their distributed capacitive effects. The method uses a finite-element mesh to first construct an RC network that models the distributed RC effects in detail. Then this complex RC network is transformed into a simple RC network, which is more tractable for verification purposes, by repeatedly eliminating nodes from the network. During each elimination step it is ensured that the Elmore or first-order time constants between the terminals of each interconnection are preserved. This guarantees that the electrical transfer function of the final network closely matches the electrical transfer function of the initial network and, consequently, that of the distributed RC interconnect. The method is efficiently applied in a layout-to-circuit extraction program which moves a scanline over the layout and executes all operations on a relatively small collection of layout objects tied to the scanline. An implementation of the method has been done in the layout-to-circuit extractor called SPACE, and experimental results are shown with respect to extraction times, memory usage, and accuracy.

In Chapter 3 the computation of three-dimensional capacitive effects of interconnections in VLSI circuits is considered. The 3-D capacitive effects of interconnections are especially important in submicron VLSI circuits since the

vertical dimensions of the interconnections in these circuits are in the same order of magnitude as the minimum feature sizes in the horizontal direction. A method is described that is based on a boundary-element technique, and that solves a 3-D capacitance model in a time that is linear with the size of the circuit and with a memory usage that is constant. The method uses a special application of the Schur algorithm to approximately solve the Green's function integral equations that result from using the boundary-element technique. Instead of inverting the full matrix that relates to the set of integral equations, the Schur algorithm finds an approximate matrix, so that (1) the complexity of the capacitance model is reduced - small capacitances between conductors that are far-apart are not computed - and (2) memory and computation complexities of the method are reduced. It is shown how the Schur algorithm is applied to efficiently solve the 3-D interconnect capacitances of an arbitrary conductor configuration, and how pipelining and parallelization of the algorithm are used to further reduce the memory requirements and the computation time of the method. Further, this method has been implemented in the layout-to-circuit extraction program SPACE to demonstrate the applicability of the method and to provide experimental results.

Finally, Chapter 4 discusses the simulation of the logic and timing behavior of large digital MOS circuits, using the models of the previous chapters. The simulation model that is presented is based on a "switch-level model" and it allows one to quickly obtain an estimate of the logic and timing behavior of a VLSI circuit, based on actual transistor and interconnection parameters. The simulation uses a transistor model in which each MOS transistor is modeled by a gate-voltage controlled switch in series with a resistor between drain and source. Each node in the network has a capacitance to ground connected to it. In addition to a logic value O, I or X that is associated with it, each node stores a set of parameters that describes the current rising or falling waveform for that node. This waveform is used to obtain a first-order approximation for the real voltage waveform, and thus to accurately represent transient effects like spikes and races. To achieve the desired simulation speed, the first-order voltage waveform approximations are computed by means of simple resistance division and charge-sharing computations and by means of first-order time-constant evaluations. Both a minimum and maximum voltage waveform are computed for each node (1) to model the logical X state, and (2) to model the influence of the variations of circuit parameters and the accuracy. The switch-level models have been incorporated in a switch-level simulator called SLS which has been used during the design of many VLSI circuits.

Samenvatting

De grote complexiteit en de hoge fabricage kosten van VLSI schakelingen maken het noodzakelijk dat het gedrag van deze schakelingen uitgebreid geverifiëerd wordt voordat wordt overgegaan tot fabricage. Behalve voor de hand liggende fouten zoals verkeerde of onvolledige verbindingen tussen verschillende gedeelten van het circuit kunnen elektrische fouten optreden die te wijten zijn aan fysische beperkingen zoals te grote potentiaal verschillen langs voedingslijnen, te grote waarden voor vertragingstijden, en onbedoelde capacitieve koppelingseffecten tussen de verbindingen. Om een ontwerp van een geïntegreerde schakeling op een efficiënte en nauwkeurige manier op deze fouten te kunnen controleren zijn modellen nodig die een lage complexiteit hebben, die voldoende nauwkeurig zijn, en die gemakkelijk kunnen worden ingebouwd in computer programma's voor computer-gestuurd ontwerp (CAD). In dit proefschrift worden drie van deze modelleringstechnieken besproken die elk een belangrijk aspect van het gedrag van VLSI schakelingen modelleren.

In hoofdstuk 2 wordt eerst het modelleren van de weerstandseffecten in de verbindingen van geïntegreerde schakelingen besproken. Behalve de weerstanden van de verbindingen zijn ook de gedistribueerde capacitieve effecten van belang bij het karakteriseren van het transmissie gedrag van de verbindingen, en daarom wordt vervolgens ook dit onderwerp behandeld. In dit proefschrift wordt een modelleringstechniek beschreven die gebaseerd is op een eindige elementen methode, en die het mogelijk maakt om RC modellen met een lage complexiteit af te leiden welke nauwkeurig de weerstandseffecten van de verbindingen en de gedistribueerde capacitieve effecten van de verbindingen representeren. De methode maakt gebruik van een opdeling in eindige elementen om eerst een RC netwerk te construeren dat de gedistribueerde RC effecten op gedetailleerd niveau modelleerd. Daarna wordt dit complexe RC netwerk getransformeerd naar een eenvoudig RC netwerk, dat beter geschikt is verificatie doeleinden, door herhaardelijk knopen uit het netwerk te elimineren. Tijdens elke eliminatie stap wordt er voor gezorgd dat de Elmore of eerste orde tijdsconstante tussen de aansluitpinnen van elke verbinding gelijk blijft. Dit garandeert dat de elektrische overdrachtsfunktie van het uiteindelijke netwerk nauwkeurig overeenkomt met de elektrische overdrachtsfunktie van het initiële netwerk en, dientengevolge, met die van de gedistribueerde RC lijn. De methode kan op een efficiënte manier worden toegepast in een layout-naar-circuit extractie programma, dat een scan-lijn over de layout laat voortbewegen, en dat alle operaties uitvoert op een relatief kleine verzameling layout objecten die aan de scan-lijn gekoppeld zijn. Een

implementatie van de methode is gedaan in de layout-naar-circuit extractor SPACE en experimentele resultaten worden getoond betreffende extractie-tijden, geheugen-gebruik en nauwkeurigheid.

In hoofdstuk 3 wordt de berekening van drie dimensionele capacitieve effecten van verbindingen in VLSI schakelingen behandeld. De 3-D capacitieve effecten van verbindingen zijn vooral van belang voor submicron VLSI schakelingen omdat de verticale afmetingen van de verbindingen in deze schakelingen in dezelfde orde van grootte zijn als de minimum afmetingen in de horizontale richting. Een methode wordt beschreven welke gebaseerd is op een grensvlak methode, en welke het mogelijk maakt een 3-D capaciteitsmodel op te lossen in een tijd die lineair is met de grootte van de schakeling, en met een constant geheugen gebruik. De methode maakt gebruikt van een speciale toepassing van het zogeheten Schur algoritme om de Greense funktie integraalvergelijkingen die afkomstig zijn van de grensvlak methode op te lossen. In plaats van het inverteren van de volledige matrix die de verzameling van integraalvergelijkingen beschrijft, vindt het Schur algoritme een benaderende inverse zodat (1) de complexiteit van het capaciteitsmodel gereduceerd wordt - kleine capaciteiten tussen geleiders die ver van elkaar verwijderd zijn worden niet berekend - en (2) geheugen en reken-complexiteit van de methode worden gereduceerd. Er wordt aangetoond hoe het Schur algoritme toegepast wordt om op een efficiënte manier de 3-D capaciteiten te berekenen voor een willekeurige configuratie van geleiders, en hoe pipelining en parallelisatie van het algoritme gebruikt kunnen worden om geheugengebruik en rekentijden verder te reduceren. Ook deze methode is geïmplementeerd in het layout-naar-circuit extractie programma SPACE om de toepasbaarheid van de methode aan te tonen en om experimentele resultaten te verkrijgen.

Hoofdstuk 4 behandelt de simulatie van grote digitale MOS schakelingen op hun logisch en timing gedrag, daarbij gebruik makende van de modelen uit de voorafgaande hoofdstukken. Het simulatie model dat wordt beschreven is gebaseerd op een zogeheten switch-level model, en maakt het mogelijk om snel een schatting van het logisch en timing gedrag van een VLSI schakeling, gebaseerd op werkelijke transistor en verbindings parameters, te verkrijgen. Het simulatie model maakt gebruik van een transistor model waarin elke MOS transistor wordt gemodelleerd door een gate-spanningsgestuurde schakelaar in serie met een weerstand tussen drain en source. Elke knoop in het netwerk heeft een capaciteit naar aarde en, naast een logische waarde O, I of X die geassociëerd wordt met elke knoop, een verzameling parameters die de op dat moment geldende stijgende of dalende spanningsgolfvorm voor die knoop beschrijven. Deze golfvorm vormt een eerste orde benadering voor de werkelijke

spanningsgolfvorm voor de knoop en, op deze manier, een nauwkerige benadering van overgangsverschijnselen zoals spikes en races. Om de gewenste simulatie snelheid te halen worden de eerste orde benaderingen van de spanningsgolfvormen berekend m.b.v. eenvoudige weerstandsdeling en ladingsdeling berekeningen en m.b.v eerste orde tijdsconstante evaluaties. Zowel een minimum als een maximum spanningsgolfvorm worden berekend voor elke knoop om (1) de logische X toestand te kunnen modelleren, en (2) de invloed van veranderingen in circuit parameters en de invloed van nauwkeurigheid te kunnen modelleren. De switch-level modellen zijn opgenomen in switch-level simulator SLS, welke praktisch gebruikt is tijdens het onwerpen van vele VLSI schakelingen.

Acknowledgements

This research was supported by the commission of the EC under Esprit contract 991 and by the Dutch FOM under IOP-IC contract 45.009. The cooperation of all those participating in these projects is gratefully acknowledged. The people of the 16th floor, Department of Electrical Engineering, Delft University of Technology, are thanked for their daily cooperation, especially my colleague Nick van der Meijs and the department secretary Corrie Boers-Boonekamp. I am indebted to many users of the software that was developed as a result of this work, for giving valuable feedback. People who have directly contributed to the work presented in this thesis are, Chapter 2: Petr Kazil and Nick van der Meijs; Chapter 3: Prof. Dewilde, Martin Lossie, Harry Nelis, Zhen-Qiu Ning and Nick van der Meijs; Chapter 4: Sander de Graaf, Oscar Hol and Alex Schooneveld. Prof. Dewilde was an inspiring organizer and leader of many activities that provided the setting for this work, and he gave many useful comments. Mrs. J.B. Zaat-Jones corrected the English text in this thesis. Finally I wish to thank my parents for their patience and their support.

Biography

Arjan van Genderen was born in Vlaardingen, the Netherlands, on August 17, 1961. After receiving his VWO diploma in 1979 from the Chr. Scholengemeenschap Westland-Zuid, Vlaardingen, he started his study of electrical engineering at Delft University of Technology, Delft, the Netherlands. He graduated (cum laude) in July 1985. In August 1985 he joined the Section of Network Theory as a research assistant. There he worked towards his Ph.D. degree, under the supervision of prof. dr. ir. P.M. Dewilde, within the project "Accurate 3-D Extraction of Circuit Components from VLSI Layout." He is the author of the switch-level program SLS, which has become an important part of the Nelsis IC design system, and a coauthor of the Nelsis layout-to-extraction program SPACE.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EXHIBITS